Protein Fold Recognition from Sequences using Convolutional and Recurrent Neural Networks

Amelia Villegas-Morcillo, Angel M. Gomez, Juan A. Morales-Cordovilla, Victoria Sanchez

Department of Signal Theory, Telematics and Communications, University of Granada, Spain
E-mail: {ameliavm, amgg, jamc, victoria}@ugr.es

Abstract – The identification of a protein fold type from its amino acid sequence provides important insights about the protein 3D structure. In this paper, we propose a deep learning architecture that can process protein residue-level features to address the protein fold recognition task. Our neural network model combines 1D-convolutional layers with gated recurrent unit (GRU) layers. The GRU cells, as recurrent layers, cope with the processing issues associated to the highly variable protein sequence lengths and so extract a fold-related embedding of fixed size for each protein domain. These embeddings are then used to perform the pairwise fold recognition task, which is based on transferring the fold type of the most similar template structure. We compare our model with several template-based and deep learning-based methods from the state-of-the-art. The evaluation results over the well-known LINDAHL and SCOP_TEST sets, along with a proposed LINDAHL test set updated to SCOP 1.75, show that our embeddings perform significantly better than these methods, specially at the fold level. Supplementary material, source code and trained models are available at http://sigmat.ugr.es/~amelia/CNN-GRU-RF+/.

Index terms – Protein Fold Recognition, Deep Learning, Convolutional Neural Networks, Recurrent Neural Networks, Embedding Learning, Random Forests

1 Introduction

The determination of the protein's tri-dimensional structure from its amino acid sequence is one of the main challenges in structural biology. Its importance is given by the close relationship between the protein's 3D structure and its biological function. Although structure prediction at atomic level is a hard problem, knowing the fold type [1] [2] the protein belongs to may disclose relevant information about its structure and function [3]. Protein fold prediction can be accomplished by comparison with related proteins from the Protein Data Bank (PDB) database [4] that are already classified in different levels according to sequential and structural similarities [5].

The Structural Classification of Proteins (SCOP) database [6] and its latest extended version SCOPe [7] provide a hierarchical classification of protein domains in structural classes, folds, superfamilies, families, proteins and species. According to SCOP, structural class and fold levels comprise domains that share structural features and similar topology, and do not imply sequence homology as those in the same superfamily or family. Moreover, it has been estimated that the number of possible folds is limited in nature [8].

State-of-the-art computational methods aim to identify protein folds following two main approaches, which are known as protein fold classification and protein fold recognition. The taxonomy-based fold classification approach [9] can be viewed as a typical multi-class classification problem, in which the protein sequences are directly mapped into fold classes. The machine learning-based methods, such as [10], FP-Pred [11], ACCFold [12], TAXFOLD [13], HMMFold [14] and ProFold [15], were designed to successfully classify into a predefined group of SCOP fold classes. However, the selected folds comprise a small set including only those folds with a higher amount of protein domains (27 folds in [10] [12] or 30 folds in [16]), in contrast to the more than one thousand existing fold classes in the SCOP database.

On the other hand, the protein fold recognition approach is derived from the template-based structure prediction problem (homology modelling and threading), where the fold type is inferred by comparing with template proteins with known structure [17] [18]. In the fold recognition methods, the query protein is compared with a set of templates and the fold class of the most similar template is transferred to the query [19]. Homology modelling methods recognize template proteins which present a high sequence similarity with the query protein. This can be assessed by sequence-to-sequence alignment [20] or profile-to-profile alignment using hidden Markov models (HHpred [21]) and Markov random fields (MRFAlign [22]). On its part, threading methods are suitable for those proteins that have low similarity in sequence, and try to match the query sequence with template structures using structural properties. Threading methods include RAPTOR [23], BoostThreader [24], SPARKS-X [25], CNFpred [26] and [27], mostly based on conditional random fields, and CEthreader [28]. More recently, the fold recognition task has been addressed as a binary classification problem for each query-template protein pair by using machine learning tools such as support vector machines (FOLDpro [29]), random forests (RF-Fold [30]) and deep neural networks (DN-Fold [31]). Furthermore, other ensemble methods that combine multiple feature types (describing amino acid sequences, evolutionary changes or structure properties) and different prediction techniques have been proposed, such as the machine learning with template-based recognition TA-Fold [32], the multi-view model MT-fold [33], and the learning to rank Fold-LTR-TCP [34] methods.

In the past few years, deep learning [35] approaches have been introduced to improve the existing protein fold recognition methods. These methods rely on training a deep neural network to classify the whole set of SCOP fold classes. Then, the outputs of the last hidden layer are extracted as a fold-related embedding vector for each protein domain, which is used to perform pairwise protein fold recognition. Among the state-of-the-art techniques we can find the deep learning methods DeepFR [36], DeepSF [37], DeepSVM-fold [38] and MotifCNN-fold [39]. The proposed architectures are mainly based on applying convolution operations over image-form features (predicted contact maps) and protein residue-level features (evolutionary and secondary structure predictions). However, convolutional networks only exploit local relations within the protein sequence. Another hurdle is that protein sequences cover a wide range of possible lengths (from 10 to 10000 amino acids), but feed-forward neural network architectures require a fixed-size input. In order to tackle these issues, we propose here a different architecture which introduces a recurrent layer after the CNN to obtain a fixed-size representation from the variable-length input sequence.

Recurrent neural networks (RNN) with long short-term memory (LSTM) units [40] have been employed in sequential problems such as speech recognition [41] and machine translation [42], as well as in the field of proteomics, addressing the protein homology detection [43] and protein secondary/tertiary structure prediction [44] [45] tasks. Moreover, architectures that combine deep CNN with LSTM have been proposed in proteomics to improve the predictions of subcellular localizations of proteins [46], residue-residue contact maps [47], protein secondary structure [48] and local backbone angles [49]. Thereby, the promising results achieved in these studies suggest that neural networks including recurrent layers can also be suitable for addressing the protein fold recognition task, as shown by the recent DeepSVM-fold [38] method. However, the neural network architecture introduced in [38] imposes limitations on the length of the protein sequences to be processed. In this paper we go one step further and propose a more straightforward architecture that can handle protein sequences of any length.

Thus, the contributions of this work are several. First of all, we propose a neural network architecture that combines convolutional and recurrent layers, which is able to process protein residue-level features of arbitrary length, and classify them into all the SCOP fold classes. Secondly, we leverage the trained neural network to extract a fold-related embedding suitable for performing pairwise fold recognition. Finally, we provide evaluation results by combining our similarity scores with other pairwise measures in a random forest classifier, which outperforms all state-of-the-art existing methods. A complete overview of our fold recognition approach is depicted in Fig. 1.

2 Materials and Methods

2.1 Feature Extraction

In this work, we represent the protein domain with variable-length L as a sequence of L vectors, each one of size 45 representing each amino acid residue in the domain sequence (Fig. 1a), as in [37]. These 45 residue-level features include information about the amino acid, evolutionary profile, secondary structure and solvent accessibility. Thus, the first 20 elements contain a one-hot vector representation of the amino acid. The profile information is given by the position-specific scoring matrix (PSSM) that contains 20 elements for each amino acid position, and is computed by PSI-BLAST [50] using the non-redundant database 'nr90' for sequence homology searching. Secondary structure and solvent accessibility information are predicted, respectively, by the SSpro and ACCpro methods from the SCRATCH suite [51]. This secondary structure is then encoded by a one-hot vector of three elements which correspond to helix, strand and loop structural classes, whereas the solvent accessibility is encoded by a one-hot vector of two elements corresponding to exposed and buried states. The resulting $L \times 45$ features for each protein domain are then used as input to the neural network model, which is trained to map it into one of the SCOP fold classes.

2.2 Convolutional-Recurrent Neural Network Model

Our neural network model architecture (Fig. 1b) is formed by a concatenation of three different types of layers: convolutional (CNN), recurrent (RNN) and fully-connected (FC). The details and purpose of each part of the network are described in the following subsections. The number of layers, filters, units per layer, and the rest of

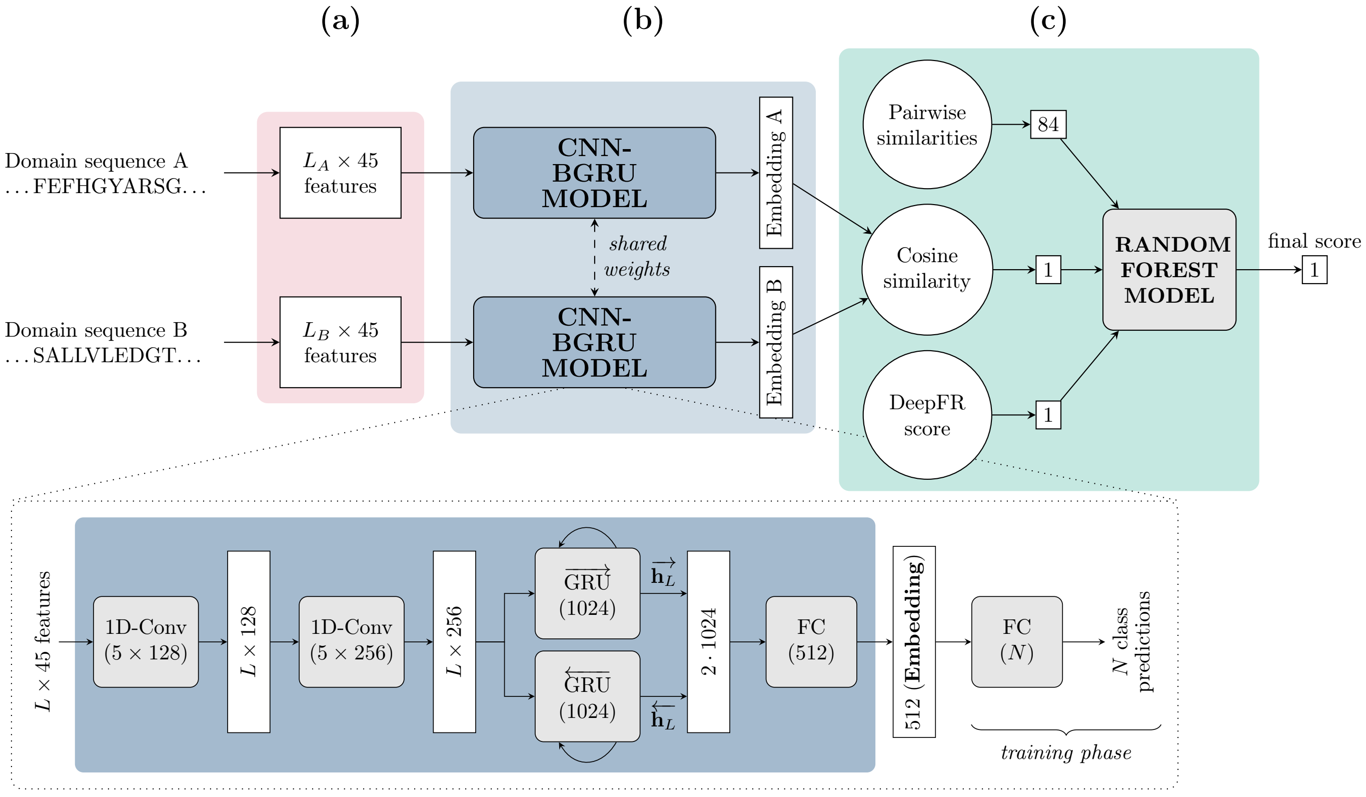

Figure 1: The proposed fold recognition approach to obtain a fold similarity score for two protein domains. **(a)** First, 45 input features are extracted for each amino acid in each of the two domain sequences. **(b)** The resulting $L \times 45$ residue-level features are passed through the CNN-BGRU model, previously trained to map protein sequences into fold classes. The model architecture (bottom part) consists of two one-dimensional convolutional layers, a bidirectional gated recurrent unit (GRU) layer and two fully-connected (FC) layers. The CNN-GRU model is very similar but with a unidirectional GRU instead and hence an output of size 1024. The two input protein domains are processed independently by the same network model (i.e. with identical trained weights). This model is then used to extract a fold-related embedding vector for each one (from the 512-dimensional output of the first FC layer). **(c)** The cosine similarity distance is computed between the two embeddings, which is concatenated to other similarity measures in a vector to obtain the final fold similarity score using a random forest model.

hyperparameters which configure the network, were chosen after conducting a cross-validation study (described in Section 2.4).

2.2.1 Convolutional neural network

The purpose of the convolutional neural network (CNN) part is to capture the local context of each amino acid in the protein domain and discover patterns in the sequence. To this end, several 1D-CNN layers are used to convolve the inputs across the sequence dimension.

In contrast to conventional 2D-CNN networks, frequently used in image processing, 1D-CNN are intended for sequence processing. In these networks, a kernel of learned weights, or convolutional filter, slides across only one dimension (instead of two). In our model, the sliding is done through the sequence dimension (from amino acid $l = 1$ to $l = L$). Thus, a filter of length k (and deep 45) is applied over the residue-level features of each amino acid l in the sequence along with those in its local context (that is, from $l - \frac{k-1}{2}$ to $l + \frac{k-1}{2}$). As each filter yields a value per amino acid and several filters are learned and applied to the same layer, the output of the 1D-CNN layer is of size $L \times K$, where K is the number of filters.

In our model, two 1D-CNN layers, using $K_1 = 128$ and $K_2 = 256$ filters of length $k_1 = k_2 = 5$, respectively, are applied over the $L \times 45$ residue-level input features (Fig. 1b). Note that the use of 1D convolutions allows the model to be insensitive to the residue-level feature size. Thus, an arbitrary number of features per amino acid could be used instead of 45 as this only implies a different deep (inner size) of the first-layer's convolutional filters. After each 1D-convolutional layer, ReLU activation and Batch-Normalization [52] are applied.

Table 1: Three levels SCOP_TEST, LINDAHL and LINDAHL_1.75 test sets for pairwise fold recognition.

Test set name	**SCOP version**	**Full test set**		**Family level**			**Superfamily level**			**Fold level**		
		Proteins	***Folds***	***Pairs***	***Proteins***	***Folds***	***Pairs***	***Proteins***	***Folds***	***Pairs***	***Proteins***	***Folds***
SCOP_TEST [31]	1.75	124	14	614	106	13	336	56	6	300	36	4
LINDAHL [19]	1.37	976	330	1646	555	121	2130	434	79	3662	321	38
LINDAHL_1.75	1.75	976	323	1532	547	120	1988	431	75	4188	356	43

2.2.2 Recurrent neural network

Unlike feed-forward neural networks, where the information only flows from the input to the output, recurrent neural networks (RNNs) introduce iteration loops that allow information to travel from the output to the input of the network as well. Thus, a typical RNN uses its own output at a previous iteration, along with the input at the current one, to compute the current output. For this reason, RNNs are frequently used for sequential data processing, exploiting long-distance relations and summarizing the sequence.

In this case, RNNs are used to iteratively process the amino acid sequence. In each iteration (i.e. for each amino acid l in the sequence), the current output is computed using the RNN output at the previous iteration (from the previous amino acid $l-1$) which, in turn, has been computed considering the output from the previous amino acid $(l-2)$ and so on. In our model, only the RNN output at the last iteration (from amino acid $l = L$) is retained as a summary of the whole domain sequence, while RNN outputs for intermediate amino acids $(l = 1, \ldots, L-1)$ are discarded.

As vanilla RNN layers suffer from the vanishing/exploding gradient problem when the sequence length L increases, gated recurrent unit (GRU) based layers, as those proposed in [53], are used in this work instead. GRU-based layers solve the gradient problem by defining and updating a state vector at each network iteration l which summarizes relevant information about the previous amino acids in the sequence $(1, \ldots, l-1)$. More information about how this state vector is updated can be found in the Supplementary Material. Thus, in each iteration, the output of a GRU layer is a vector of size H, where H is the number of state units in the GRU cell. Other authors propose retaining and post-processing the full sequence of state vectors along iterations (i.e. a matrix of $L \times H$ outputs) [38], which makes the model architecture dependent on the sequence length dimension. By contrast, as we mentioned above, in this work we only consider the last state vector $(\mathbf{h}_L)$, after processing the entire sequence of length L. This fixed-size vector represents the whole variable-length protein domain sequence. In this way, the proposed model can process domain sequences of any arbitrary length.

Therefore, in our neural network architecture, the $L \times 256$ output from the previous 1D-CNN is fed into a GRU-based recurrent layer with $H = 1024$ state units, yielding a fixed-size vector of size 1024 (Fig. 1b). We refer to this approach as CNN-GRU model. As we do not want to make assumptions about directionality relevance in protein sequence processing, the same sequence but in reverse order (i.e. from $l = L$ to $l = 1$), is also fed into another same-size GRU-based recurrent layer [54]. The outputs from both forward $(\overrightarrow{\mathbf{h}_L})$ and backward $(\overleftarrow{\mathbf{h}_L})$ GRU layers are then concatenated into a vector of 2048 elements. We refer to the model resulting from this bidirectional GRU as CNN-BGRU model.

2.2.3 Fully-connected layer

Finally, a fully-connected (FC) layer is used to learn a non-linear combination of the GRU output vector ($\overrightarrow{\mathbf{h}_L}$ of size 1024 in the CNN-GRU model or $[\overrightarrow{\mathbf{h}_L}, \overleftarrow{\mathbf{h}_L}]$ of size 2048 in the CNN-BGRU one) and create a fold-related embedding representing the protein domain. This FC layer consists of a dense layer with 512 units, after which the sigmoid activation function is applied.

While fold-related embeddings are the actual outputs of our model and the ones used for the fold recognition task during the test phase (Fig. 1c), model training requires some additional steps to be successfully accomplished. Thus, to train the network for learning suitable fold-related embeddings, a direct fold classification task is employed. To this end, a second FC layer is added to the model (Fig. 1b, bottom right). This dense layer contains N output units and performs a simple linear combination of the 512-dimensional embedding vector. Here, N is the number of fold classes in which the input proteins are going to be classified during training. The cross-entropy computed from model predictions and one-hot vectors encoding the true fold class is used as the loss function which guides the optimization process.

2.3 Datasets

In our study, we use the same set of protein domains selected from the SCOPe version 2.06 database as in [36] to train the models. This training set includes 16133 domains with less than 95% pairwise sequence identity covering $N = 1154$ folds. In order to test the method, we considered the well-known SCOP_TEST [31] and LINDAHL [19] sets, which have been widely used in previous studies for assessing the pairwise fold recognition task. The SCOP_TEST set contains 124 protein domains from SCOP version 1.75, which cover 14 fold classes, while the LINDAHL set has 976 domains from SCOP version 1.37 covering a total of 330 folds. The maximum sequence identity inside both test sets and also regarding the training set is 40%. In order to test the performance of the methods on the pairwise fold recognition problem, the protein domains within each test set are paired and evaluated at family, superfamily and fold levels, as detailed in Section 2.5. The number of positive pairs at each level and the resulting individual domains to be evaluated are reported in Table 1. As can be seen, in the SCOP_TEST set, 106, 56 and 36 domains are evaluated at family, superfamily and fold levels respectively, whereas in the LINDAHL set, we evaluate 555, 434 and 321 domains at each level respectively.

In addition, it should be mentioned that the original LINDAHL dataset is quite old (1999) and the SCOP definition of folds has changed since then. Therefore, in this paper we also propose an update of the original LINDAHL from SCOP version 1.37 to version 1.75, where the family, superfamily and fold classes are consistent with the most recent versions of SCOPe. To do so, we searched for the original LINDAHL sequences in the SCOP v1.75 database and then we updated the modified domain identifiers and class labels. In Table 1, we show the total number of fold classes included in the new LINDAHL_1.75 set. As can be observed, the number of pairs and protein domains at each level have changed, so the evaluation at fold level is now carried out over more domains covering more fold classes.

2.4 Model Training and Validation

In order to decide the best hyperparameters for our CNN-GRU and CNN-BGRU models, we performed a 5-stage cross-validation over the training set. To ensure independence between cross-validation subsets, we split the family classes contained within each structural class into 5 groups. Next, we used 4 subsets to train the model and the remaining one to validate it. Tested hyperparameter values included model architecture configuration, learning rate and number of epochs. We selected those that achieved the highest average results to carry out a final training procedure comprising the entire training dataset.

During the training phase, the variable-length sequence domains were grouped into mini-batches of 50 samples applying zero-padding to the maximum length within each mini-batch. In the GRU layer, we kept the last state vector of each domain sample in the mini-batch (i.e. before zero-padding). We trained the network by minimizing the cross-entropy loss function for a fixed number of epochs. We used the Adam optimizer [55] with an initial learning rate of 10^{-3}, which we reduced to 10^{-4} in epoch number 40 as cross-validation suggested. The whole optimization process was completed in 80 epochs. In order to prevent overfitting to the most populated fold classes, we applied L2 penalty with a small weight decay of $5 \cdot 10^{-4}$, and dropout [56] with a drop probability of 0.2 in the CNN and the first FC layers.

Our neural network models were implemented using Pytorch [57] (version 1.0) and trained on a GPU card (NVIDIA GTX Titan X, 12GB). The GPU memory required for training our CNN-GRU and CNN-BGRU models was 4.0 and 6.6 GB, respectively. The network complexity was estimated in terms of MAC (multiply-accumulator unit) counts, yielding 2.4 MMACs per processed amino acid for the CNN-GRU model and 3.5 MMACs for the CNN-BGRU model. Moreover, the CNN-GRU and CNN-BGRU models needed 75 and 109 seconds per epoch, completing the 80 training epochs in approximately 2 and 3 hours, respectively.

2.5 Evaluation and Similarity Measures

Evaluation is conducted in terms of pairwise fold recognition accuracy, using the fold-related embedding vectors. In this task, all the test domains are paired and evaluated following the SCOP hierarchy at three levels with increasing difficulty, as proposed by [19]: (i) *family level*, populated with pairs of test domains that share the same family class, (ii) *superfamily level*, with pairs of test domains that share the same superfamily class, but not the same family, and (iii) *fold level*, with pairs of test domains that share the same fold class, but neither share the same family nor superfamily.

At each level, the individual protein domains are considered as queries which are compared to a pool of templates. We addressed this task in the same way as the DeepFR method [36]. To compare each pair of query-template domains, we first use the extracted fold-related embeddings from each neural network model (CNN-GRU or CNN-BGRU). This comparison is carried out by computing the cosine similarity distance between the query and the template embedding vectors (Fig. 1c). Then, the templates for each query are ranked by score value and, following the nearest neighbor condition, the fold class of the most similar template is assigned to the query domain. We refer to these methods as CNN-GRU and CNN-BGRU.

This first fold similarity score was enhanced by training a random forest (RF) model using our cosine similarity score along with the 84 similarity measures used by the DeepFRpro method [36]. These pairwise measures contain sequence similarity information, sequence and profile alignment features, and structural relations [30, 31]. Thus, the RF input vectors are of size 85 and correspond to each pair of proteins. As before, we used the RF output pairwise scores to perform template ranking for each query domain. We named these methods CNN-GRU-RF and CNN-BGRU-RF, depending on the recurrent network used for extracting the fold-related embeddings (unidirectional or bidirectional).

Finally, with the aim of including structural information from predicted contact maps, we also concatenated the cosine similarity score from the DeepFR method to our vector of pairwise scores (total size of 86). This extended score vector was then used to train a second random forest model and obtain an improved fold similarity score (Fig. 1c, CNN-GRU-RF+ and CNN-BGRU-RF+ methods).

The random forest classifiers were trained with 500 decision trees, using the Python Scikit-learn package [58] implementation. As in [36], for the SCOP_TEST set the RF models were trained on the whole LINDAHL set, while we performed a 10-stage cross-validation to evaluate the LINDAHL and LINDAHL_1.75 test sets.

The performance metric used in all approaches is the fold recognition accuracy, which computes the ratio of samples that have been correctly classified (top 1 prediction). We also provide the ratio of finding the true fold in one of the five classes with the highest score values (named as top 5 prediction).

3 Results and Discussion

The experimental accuracy results for the SCOP_TEST and LINDAHL test sets can be found in Table 2 and Table 3, respectively. We provide pairwise fold recognition results at the family, superfamily and fold levels, considering both the top 1 and top 5 predictions, as described in Section 2.5. In both test sets, we compare our CNN-GRU (CNN-GRU-RF, CNN-GRU-RF+) and CNN-BGRU (CNN-BGRU-RF, CNN-BGRU-RF+) based methods to several template-based and deep learning-based methods from the state-of-the-art that also address the protein fold recognition problem.

For the SCOP_TEST set (Table 2), our CNN-BGRU method obtained the best results at fold level, with accuracy values of 91.7% (33/36 hits) and 94.4% (34/36 hits) at top 1 and top 5 predictions, respectively. These results were closely followed by our CNN-BGRU-RF and CNN-BGRU-RF+ methods, both achieving 88.9% accuracy at the fold level (91.7% at top 5). The mentioned methods also provided good results at the family and superfamily levels. Indeed, our CNN-BGRU-RF+ method yielded the best accuracy values at superfamily level (92.9% and 96.4% at top 1 and top 5 predictions). Furthermore, as can be seen in Table 3, our CNN-BGRU-RF+ method outperformed all state-of-the-art methods at the fold level considering the LINDAHL test set. It achieves accuracy values of 76.3% (245/321 hits) and 85.7% (275/321 hits) at top 1 and top 5 predictions, respectively. This method was closely followed by our CNN-GRU-RF+ method, which also yielded the best results at the superfamily level (78.3% and 88.0% at top 1 and top 5 predictions).

In order to assess the raw performance of the fold-related embeddings extracted from our CNN-GRU and CNN-BGRU models, we can compare our results with those obtained from the DeepFR method [36] (both strategies 1 and 2). In this case, the metric used for embedding comparison is the cosine similarity distance. A notable difference is that our models use protein residue-level features at input, while the DeepFR model processes more informative structural features in the form of predicted contact maps. The evaluation results showed that our models performed better than the best DeepFR method (strategy 2). Specifically, our CNN-BGRU model stands out for its top 1 and top 5 predictions in both test sets (Tables 2 and 3). These results imply that our fold-related embedding vectors are suitable for template ranking and fold matching. Our models succeed in processing the input protein residue-level features (which include evolutionary and secondary structure information). The performance is comparable or even better to that obtained with image-form contact map features by the DeepFR method. This can be attributed to the use of recurrent layers that properly cope with the variable-length protein sequence, providing a well-suited fold-related embedding.

To further explore whether the extracted embeddings from our CNN-BGRU model are related to the fold classes, we run bi-clustering over a subset of SCOP_TEST with 46 protein domains covering 6 folds. These fold classes are $a.3$, $b.36$, $c.1$, $c.67$, $d.41$, and $f.4$, which provided better classification results in the SCOP_TEST dataset using the CNN-BGRU model. Fig. 2 shows the resulting dendroheatmap, where each row corresponds to the 512-dimensional embedding extracted for each sample. As can be seen, protein domains in the same fold class present similar blocks (in red and green colors) in their embedding vectors. Also, the hierarchical clustering differentiates 6 clusters, each one grouping protein domains from the same fold together. We notice that folds $c.1$ and $c.67$ (from structural class c) cluster together at a higher level, indicating that the clustering is consistent with the SCOPe hierarchy. These findings suggest that the extracted embedding vectors are fold related.

We then evaluated the pairwise score obtained from our random forest classifier (using the 85-dimensional score vectors) in the fold recognition task. The results on the LINDAHL test set show an improvement of our

Table 2: Pairwise fold recognition accuracy results at the family, superfamily and fold levels within the SCOP_TEST set.

Method	Family		Superfamily		Fold	
	Top 1	***Top 5***	***Top 1***	***Top 5***	***Top 1***	***Top 5***
RF-Fold [31]	93.4	98.1	83.9	91.1	55.6	72.2
DN-Fold [31]	94.3	97.2	82.1	91.1	61.1	86.1
RFDN-Fold [31]	93.4	97.2	82.1	91.1	61.1	86.1
MRFalign [36]	91.5	98.1	85.7	94.6	63.9	72.2
CEthreader [28]	84.0	95.3	73.2	89.3	80.6	91.7
DeepFR (s1) [36]	74.5	91.5	76.8	87.5	77.8	80.6
DeepFR (s2) [36]	75.5	93.4	78.6	83.9	86.1	88.9
DeepFRpro (s1) [36]	**95.3**	**99.1**	89.3	92.9	75.0	91.7
DeepFRpro (s2) [36]	94.3	96.2	87.5	94.6	83.3	88.9
CNN-GRU	84.9	95.3	80.4	87.5	72.2	86.1
CNN-BGRU	84.9	96.2	87.5	91.1	**91.7**	**94.4**
CNN-GRU-RF	93.4	96.2	87.5	**96.4**	75.0	83.3
CNN-BGRU-RF	91.5	95.3	87.5	92.9	88.9	91.7
CNN-GRU-RF+	93.4	96.2	89.3	94.6	83.3	88.9
CNN-BGRU-RF+	92.5	95.3	**92.9**	**96.4**	88.9	91.7

Table 3: Pairwise fold recognition accuracy results at the family, superfamily and fold levels within the LINDAHL test set.

Method	Family		Superfamily		Fold	
	Top 1	***Top 5***	***Top 1***	***Top 5***	***Top 1***	***Top 5***
PSI-BLAST [19]	71.2	72.3	27.4	27.9	4.0	4.7
HHpred [24]	82.9	87.1	58.0	70.0	25.2	39.4
RAPTOR [24]	**86.6**	89.3	56.3	69.0	38.2	58.7
BoostThreader [24]	86.5	90.5	66.1	76.4	42.6	57.4
SPARKS-X [25]	84.1	90.3	59.0	76.3	45.2	67.0
FOLDpro [31]	85.0	89.9	55.0	70.0	26.5	48.3
RF-Fold [31]	84.5	91.5	63.4	79.3	40.8	58.3
DN-Fold [31]	84.5	91.2	61.5	76.5	33.6	60.7
RFDN-Fold [31]	84.7	91.5	65.7	78.8	37.7	61.7
TA-fold [32]	85.2	—	74.2	—	53.9	—
MRFalign [36]	85.2	90.8	72.4	80.9	38.6	56.7
CEthreader [28]	76.6	87.2	69.4	81.8	52.3	70.4
DeepFR (s1) [36]	67.4	80.9	47.0	63.4	44.5	62.9
DeepFR (s2) [36]	65.4	83.4	51.4	67.1	56.1	70.1
DeepFRpro (s1) [36]	85.6	91.9	66.6	82.0	57.6	73.8
DeepFRpro (s2) [36]	83.1	92.3	69.6	82.5	66.0	78.8
MT-fold [33]	—	—	—	—	54.1	—
DeepSVM-fold [38]	—	—	—	—	67.3	—
MotifCNN-fold [39]	—	—	—	—	72.6	—
Fold-LTR-TCP [34]	—	—	—	—	73.2	—
CNN-GRU	68.6	89.2	56.2	77.4	56.7	74.1
CNN-BGRU	71.0	87.7	60.1	77.2	58.3	78.8
CNN-GRU-RF	84.9	94.1	74.4	**88.5**	63.9	81.3
CNN-BGRU-RF	85.6	93.0	72.4	86.9	65.4	82.2
CNN-GRU-RF+	84.5	**95.0**	**78.3**	88.0	73.2	**86.3**
CNN-BGRU-RF+	85.4	93.5	73.3	87.8	**76.3**	85.7

advanced methods CNN-GRU-RF and CNN-BGRU-RF with respect to using our cosine similarity score alone. Accuracy results are competitive when compared to other previous template-based methods (as RAPTOR [23]

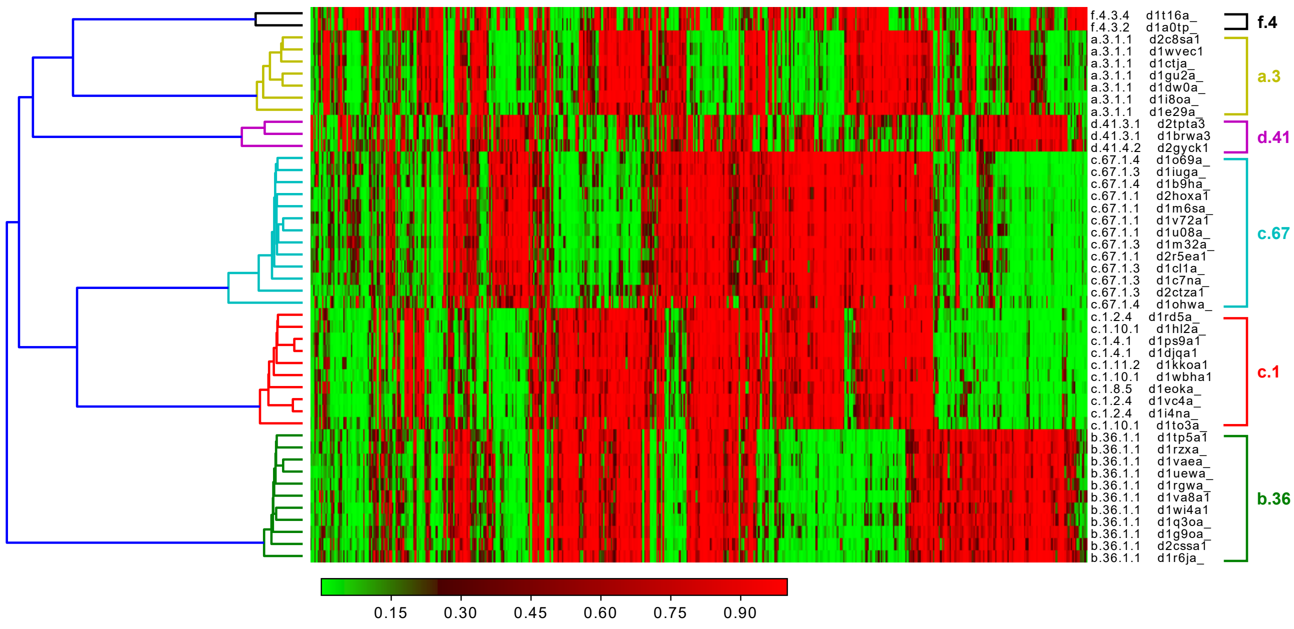

Figure 2: Dendroheatmap of the 512-dimensional fold-related embeddings extracted from the CNN-BGRU model. The analysis has been done by running bi-clustering over 46 protein domains (covering 6 folds) in the SCOP_TEST set. We computed the cosine distance between embedding vectors (rows) and embedding components (columns) separately. We then applied hierarchical clustering with single linkage to group similar vectors and components together. The embedding components are colored according to their values (lower values in green and higher values in red).

and RF-Fold [30]) and ensemble methods (as TA-fold [32] and DeepFRpro [36]) at the three levels. Our methods also provide good results if we consider the top 5 predictions, outperforming the most similar method DeepFRpro (both s1 and s2) at the three levels. When evaluating the smaller SCOP_TEST set, we noticed a greater improvement from our CNN-GRU-RF method than the CNN-BGRU-RF version, likely due to its good results before the integration of different scores.

Our CNN-GRU-RF+ and CNN-BGRU-RF+ methods led to a further performance in both test sets, specially at the superfamily and fold levels. In this case, we integrate pairwise scores derived from both evolutionary information and predicted contact maps, so that our LINDAHL results can be fairly compared with those from the recent deep learning-based methods DeepSVM-fold [38] and MotifCNN-fold [39]. We can see that our CNN-GRU-RF+ and CNN-BGRU-RF+ methods performed better than these methods at the fold level. Moreover, our CNN-BGRU-RF+ method outperformed the best method from the state-of-the-art, Fold-LTR-TCP [34]. These results suggest that our fold-related embeddings contain complementary information to those of the DeepFR method (derived from predicted contact maps), confirming the importance of combining both sequential and structural information for the protein fold recognition task.

The pairwise fold recognition was also assessed using the proposed LINDAHL_1.75, whose results are shown in Table 4. In this case, the DeepFR and DeepFRpro methods (both strategies 1 and 2) were trained and evaluated using the code and guidelines provided by the authors in [36]. We also extracted the 84 pairwise similarity measures for the new LINDAHL_1.75 and trained the random forest models. As can be seen in Table 4, the performance of the deep learning methods on the updated version of LINDAHL is similar to the original one at the family, superfamily and fold levels. Our CNN-GRU-RF+ provided the best top 1 results at the three levels, with accuracy values of 89.0% (487/547 hits), 77.3% (333/431 hits) and 73.6% (262/356 hits), respectively. In addition to this, our CNN-BGRU-RF+ method performed the best at the fold level considering the top 5 predictions, yielding an accuracy of 86.8% (309/356 hits). As can be seen, the updated LINDAHL_1.75 test set proposed in this work could be considered as a valid replacement of the original LINDAHL in future protein fold recognition studies.

4 Conclusion

In this study, we have proposed a deep learning method to address the protein fold recognition problem. Our neural network architecture combines convolutional and recurrent (unidirectional and bidirectional) layers in order to seamlessly process arbitrary-length protein sequences. In addition to properly processing the input residue-level features of proteins of any length, our method is able to learn fixed-size fold-related embeddings

Table 4: Pairwise fold recognition accuracy results at the family, superfamily and fold levels within the LINDAHL_1.75 test set.

Method	Family		Superfamily		Fold	
	Top 1	*Top 5*	*Top 1*	*Top 5*	*Top 1*	*Top 5*
DeepFR (s1)	69.7	82.3	43.4	60.6	40.2	59.0
DeepFR (s2)	72.2	85.6	46.6	64.3	50.8	67.1
DeepFRpro (s1)	88.3	**94.3**	71.9	82.8	56.5	74.2
DeepFRpro (s2)	87.0	93.8	71.9	82.6	63.5	77.2
CNN-GRU	70.2	88.7	55.7	77.0	57.9	77.0
CNN-BGRU	73.1	88.3	60.1	74.9	60.1	78.7
CNN-GRU-RF	87.9	93.6	74.9	87.7	69.1	80.6
CNN-BGRU-RF	87.9	93.1	71.7	87.0	66.3	83.4
CNN-GRU-RF+	**89.0**	**94.3**	**77.3**	**89.1**	**73.6**	84.0
CNN-BGRU-RF+	88.5	**94.3**	74.0	86.3	71.1	**86.8**

to perform pairwise fold recognition through similarity distance between embedding vectors. The inclusion of recurrent layers yielded very competitive results in comparison with the state-of-the-art methods, such as DeepFR, DeepSVM-fold and MotifCNN-fold, which make use of more informative structural features through predicted contact maps. In general, the learned embeddings from our CNN-BGRU model provide better results than those of our CNN-GRU model. Performance variability could be explained by the high number of parameters to be trained in the bidirectional GRU layer, which almost doubles those of the unidirectional one. Our study confirmed that the integration of both sequential and structural features is beneficial for protein fold recognition. When combining our cosine similarity score with other pairwise similarity measures, including the DeepFR score, we noticed a significant performance improvement with respect to other state-of-the-art ensemble methods. Our CNN-BGRU-RF+ method yielded the best fold level result in the LINDAHL test set, outperforming the best method Fold-LTR-TCP. We should also highlight our remarkable top 5 results in all cases, meaning that the proposed method is suitable for fold matching and template ranking. Furthermore, we provided a curated version of the well-known LINDAHL dataset updated to the SCOP version 1.75 database, which can be used as a benchmark test set in future studies.

As future work, the fold-related embeddings could be used to efficiently find templates in the template-based modeling of protein structures. Also, the proposed neural network architectures could be extended to integrate and process other protein-level features along with the residue-level features used here. These models could be further tested in the direct fold classification task, training and evaluating them in the complete set of SCOPe folds.

Acknowledgments

This work was supported by the Spanish MINECO/FEDER Project TEC2016-80141-P and the associated FPI grant BES-2017-079792. The authors also acknowledge the support of NVIDIA Corporation with the donation of a Titan X GPU.

References

[1] C. Chothia and A. V. Finkelstein, “The classification and origins of protein folding patterns,” *Annual Review of Biochemistry*, vol. 59, no. 1, pp. 1007–1035, 1990.

[2] D. T. Jones, W. R. Taylor, and J. M. Thornton, “A new approach to protein fold recognition,” *Nature*, vol. 358, no. 6381, p. 86, 1992.

[3] R. Kolodny, L. Pereyaslavets, A. O. Samson, and M. Levitt, “On the universe of protein folds,” *Annual Review of Biophysics*, vol. 42, pp. 559–582, 2013.

[4] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, “The protein data bank,” *Nucleic Acids Research*, vol. 28, no. 1, pp. 235–242, 2000.

[5] C. Hadley and D. T. Jones, “A systematic comparison of protein structure classifications: SCOP, CATH and FSSP,” *Structure*, vol. 7, no. 9, pp. 1099–1112, 1999.

[6] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, "SCOP: A structural classification of proteins database for the investigation of sequences and structures," *Journal of Molecular Biology*, vol. 247, no. 4, pp. 536–540, 1995.

[7] N. K. Fox, S. E. Brenner, and J.-M. Chandonia, "SCOPe: Structural classification of proteins–extended, integrating SCOP and ASTRAL data and classification of new structures," *Nucleic Acids Research*, vol. 42, no. D1, pp. D304–D309, 2014.

[8] R. D. Schaeffer and V. Daggett, "Protein folds and protein folding," *Protein Engineering, Design & Selection*, vol. 24, no. 1-2, pp. 11–19, 2010.

[9] L. Wei and Q. Zou, "Recent progress in machine learning-based methods for protein fold recognition," *International Journal of Molecular Sciences*, vol. 17, no. 12, p. 2118, 2016.

[10] C. H. Q. Ding and I. Dubchak, "Multi-class protein fold recognition using support vector machines and neural networks," *Bioinformatics*, vol. 17, no. 4, pp. 349–358, 2001.

[11] H.-B. Shen and K.-C. Chou, "Ensemble classifier for protein fold pattern recognition," *Bioinformatics*, vol. 22, no. 14, pp. 1717–1722, 2006.

[12] Q. Dong, S. Zhou, and J. Guan, "A new taxonomy-based protein fold recognition approach based on autocross-covariance transformation," *Bioinformatics*, vol. 25, no. 20, pp. 2655–2662, 2009.

[13] J.-Y. Yang and X. Chen, "Improving taxonomy-based protein fold recognition by using global and local features," *Proteins: Structure, Function, and Bioinformatics*, vol. 79, no. 7, pp. 2053–2064, 2011.

[14] J. Lyons, A. Dehzangi, R. Heffernan, Y. Yang, Y. Zhou, A. Sharma, and K. Paliwal, "Advancing the accuracy of protein fold recognition by utilizing profiles from hidden Markov models," *IEEE Transactions on Nanobioscience*, vol. 14, no. 7, pp. 761–772, 2015.

[15] D. Chen, X. Tian, B. Zhou, and J. Gao, "ProFold: Protein fold classification with additional structural features and a novel ensemble classifier," *BioMed Research International*, vol. 2016, pp. 1–10, 2016.

[16] Y. H. Taguchi and M. M. Gromiha, "Application of amino acid occurrence for discriminating different folding types of globular proteins," *BMC Bioinformatics*, vol. 8, no. 1, p. 404, 2007.

[17] J. Chen, M. Guo, X. Wang, and B. Liu, "A comprehensive review and comparison of different computational methods for protein remote homology detection," *Briefings in Bioinformatics*, vol. 19, no. 2, pp. 231–244, 2016.

[18] M. S. Abual-Rub and R. Abdullah, "A survey of protein fold recognition algorithms," *Journal of Computer Science*, vol. 4, no. 9, pp. 768–776, 2008.

[19] E. Lindahl and A. Elofsson, "Identification of related proteins on family, superfamily and fold level," *Journal of Molecular Biology*, vol. 295, no. 3, pp. 613–625, 2000.

[20] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, no. 3, pp. 403–410, 1990.

[21] J. Söding, "Protein homology detection by HMM–HMM comparison," *Bioinformatics*, vol. 21, no. 7, pp. 951–960, 2005.

[22] J. Ma, S. Wang, Z. Wang, and J. Xu, "MRFalign: Protein homology detection through alignment of Markov random fields," *PLoS Computational Biology*, vol. 10, no. 3, p. e1003500, 2014.

[23] J. Xu, M. Li, D. Kim, and Y. Xu, "RAPTOR: optimal protein threading by linear programming," *Journal of Bioinformatics and Computational Biology*, vol. 1, no. 1, pp. 95–117, 2003.

[24] J. Peng and J. Xu, "Boosting protein threading accuracy," in *Annual International Conference on Research in Computational Molecular Biology*, 2009, pp. 31–45.

[25] Y. Yang, E. Faraggi, H. Zhao, and Y. Zhou, "Improving protein fold recognition and template-based modeling by employing probabilistic-based matching between predicted one-dimensional structural properties of query and corresponding native properties of templates," *Bioinformatics*, vol. 27, no. 15, pp. 2076–2082, 2011.

[26] J. Ma, J. Peng, S. Wang, and J. Xu, "A conditional neural fields model for protein threading," *Bioinformatics*, vol. 28, no. 12, pp. i59–i66, 2012.

[27] J. A. Morales-Cordovilla, V. Sanchez, and M. Ratajczak, "Protein alignment based on higher order conditional random fields for template-based modeling," *PLoS ONE*, vol. 13, no. 6, p. e0197912, 2018.

[28] W. Zheng, Q. Wuyun, Y. Li, S. Mortuza, C. Zhang, R. Pearce, J. Ruan, and Y. Zhang, "Detecting distant-homology protein structures by aligning deep neural-network based contact maps," *PLoS Computational Biology*, vol. 15, no. 10, pp. 1–27, 2019.

[29] J. Cheng and P. Baldi, "A machine learning information retrieval approach to protein fold recognition," *Bioinformatics*, vol. 22, no. 12, pp. 1456–1463, 2006.

[30] T. Jo and J. Cheng, "Improving protein fold recognition by random forest," *BMC Bioinformatics*, vol. Vol. 15, no. 11, p. S14, 2014.

[31] T. Jo, J. Hou, J. Eickholt, and J. Cheng, "Improving protein fold recognition by deep learning networks," *Scientific Reports*, vol. 5, p. 17573, 2015.

[32] J. Xia, Z. Peng, D. Qi, H. Mu, and J. Yang, "An ensemble approach to protein fold classification by integration of template-based assignment and support vector machine classifier," *Bioinformatics*, vol. 33, no. 6, pp. 863–870, 2016.

[33] K. Yan, X. Fang, Y. Xu, and B. Liu, "Protein fold recognition based on multi-view modeling," *Bioinformatics*, vol. 35, no. 17, pp. 2982–2990, 2019.

[34] B. Liu, Y. Zhu, and K. Yan, "Fold-LTR-TCP: protein fold recognition based on triadic closure principle," *Briefings in Bioinformatics*, 2019.

[35] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[36] J. Zhu, H. Zhang, S. C. Li, C. Wang, L. Kong, S. Sun, W.-M. Zheng, and D. Bu, "Improving protein fold recognition by extracting fold-specific features from predicted residue–residue contacts," *Bioinformatics*, vol. 33, no. 23, pp. 3749–3757, 2017.

[37] J. Hou, B. Adhikari, and J. Cheng, "DeepSF: deep convolutional neural network for mapping protein sequences to folds," *Bioinformatics*, vol. 34, no. 8, pp. 1295–1303, 2018.

[38] B. Liu, C.-C. Li, and K. Yan, "DeepSVM-fold: protein fold recognition by combining support vector machines and pairwise sequence similarity scores generated by deep learning networks," *Briefings in Bioinformatics*, 2019.

[39] C.-C. Li and B. Liu, "MotifCNN-fold: protein fold recognition based on fold-specific features extracted by motif-based convolutional neural networks," *Briefings in Bioinformatics*, 2019.

[40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[41] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *International Conference on Machine Learning*, 2014, pp. 1764–1772.

[42] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 3104–3112.

[43] S. Hochreiter, M. Heusel, and K. Obermayer, "Fast model-based protein homology detection without alignment," *Bioinformatics*, vol. 23, no. 14, pp. 1728–1736, 2007.

[44] S. K. Sønderby and O. Winther, "Protein secondary structure prediction with long short term memory networks," *arXiv preprint arXiv:1412.7828*, 2014.

[45] M. AlQuraishi, "End-to-end differentiable learning of protein structure," *Cell Systems*, vol. 8, no. 4, pp. 292–301, 2019.

[46] S. K. Sønderby, C. K. Sønderby, H. Nielsen, and O. Winther, "Convolutional LSTM networks for subcellular localization of proteins," in *International Conference on Algorithms for Computational Biology*. Springer, 2015, pp. 68–80.

[47] J. Hanson, K. Paliwal, T. Litfin, Y. Yang, and Y. Zhou, "Accurate prediction of protein contact maps by coupling residual two-dimensional bidirectional long short-term memory with convolutional neural networks," *Bioinformatics*, vol. 34, no. 23, pp. 4039–4045, 2018.

[48] B. Zhang, J. Li, and Q. Lü, "Prediction of 8-state protein secondary structures by a novel deep learning architecture," *BMC Bioinformatics*, vol. 19, no. 1, p. 293, 2018.

[49] J. Hanson, K. Paliwal, T. Litfin, Y. Yang, and Y. Zhou, "Improving prediction of protein secondary structure, backbone angles, solvent accessibility and contact numbers by using predicted contact maps and an ensemble of recurrent and residual convolutional neural networks," *Bioinformatics*, pp. 1–8, 2018.

[50] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs," *Nucleic Acids Research*, vol. 5, no. 17, pp. 3389–3402, 1997.

[51] C. N. Magnan and P. Baldi, "SSpro/ACCpro 5: almost perfect prediction of protein secondary structure and relative solvent accessibility using profiles, machine learning and structural similarity," *Bioinformatics*, vol. 30, no. 18, pp. 2592–2597, 2014.

[52] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *International Conference on Machine Learning*, pp. 448–456, 2015.

[53] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[54] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[56] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[57] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[58] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.