Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 18(4): 3404–3422.
DOI: 10.3934/mbe.2021170
Received: 20 February 2021
Accepted: 07 April 2021
Published: 16 April 2021

Research article

Secondary structure prediction of protein based on multi scale convolutional attention neural networks

Ying Xu and Jinyong Cheng*

School of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, China

* **Correspondence:** Email: cjy@qlu.edu.cn.

Abstract: To fully extract the local and long-range information of amino acid sequences and enhance the effective information, this research proposes a secondary structure prediction model of protein based on a multi-scale convolutional attentional neural network. The model uses a multi-channel multi-scale parallel architecture to extract amino acid structure features of different granularity according to the window size. The reconstructed feature maps are obtained via multiple convolutional attention blocks. Then, the reconstructed feature map is fused with the input feature map to obtain the enhanced feature map. Finally, the enhanced feature map is fed to the Softmax classifier for prediction. While the traditional cross-entropy loss cannot effectively solve the problem of non-equilibrium training samples, a modified correlated cross-entropy loss function may alleviate this problem. After numerous comparison and ablation experiments, it is verified that the improved model can indeed effectively extract amino acid sequence feature information, alleviate overfitting, and thus improve the overall prediction accuracy.

Keywords: convolutional neural network; classification; attention mechanism; cross entropy loss; multi channel ; protein

1. Introduction

Proteins not only provide the material for basic life activity, but they also provide the theoretical basis for "attack of disease" theories [1]. In proteins, 85% of amino acid residues are in three basic secondary structure states: alpha-helix, beta-fold, and random coiling. Additionally a small percentage of beta-turns are also basic secondary structure states. The composition of secondary protein structures are highly regular; these structures are also non-uniformly distributed throughout the protein [2]. The accurate prediction of secondary structures can provide useful information for protein disorder prediction [3] and protein tertiary structure prediction [4]. The secondary structure of

Protein also helps to identify protein functional domains and can guide the rational design of site-specific mutagenesis experiments.

Prediction of secondary structures of protein originated in the last century, mainly via amino acid sequences. For example, Rost et al. [5] proposed a PHD-based protein prediction algorithm that used evolutionary information contained in multiple sequence alignments as input to a neural network, the accuracy of predicting the secondary structures of the protein was over 70%. Later, more information on protein features was added by capturing local and long-range protein information. Prediction studies were also carried out by combining support vector machines [6, 7], Markov models [8], Bayesian, and other statistical methods. Kurniawan et al. [9] combined PSSM features and physicochemical features of protein structures with SVM for prediction with an accuracy of about 80%. In [10], the authors combined PSSM features, hydrophobic sequence features (HSF) and structural sequence features (SSF) to obtain an overall accuracy of 78%. Zhang et al. [11] proposed combining motif encoding with PSSM features and using an autoencoder for protein prediction, and finally demonstrated that the method was 1.39% higher than using the autoencoder alone. In [7], the authors combined 21 PSSM features and 4 amino acid physicochemical property features to transform the input features into a $25 \times L$ (L is the length of the protein) feature matrix, which was input into a hybrid SVM structure for prediction. The method obtained an accuracy of 85.58% in CB513 data. However, these methods require manual construction of task-related features, which are not universal and require high labor costs.

In recent years, deep learning techniques have received increasing attention from researchers due to their properties such as powerful data fitting ability and automatic feature generation [12, 13]. Secondary structure prediction based on deep learning has become a current hot research topic. For example, PSIPRED [14] is a popular framework for predicting secondary structures based on neural networks. It uses a PSSM matrix as the input along with two model frameworks. The fully connected neural network of each stage contains a hidden layer. PSIPRED combines the PSSM matrix and the neural network; the performance of the method exceeds the expectations, but because it only uses a simple fully connected neural network, the model framework is too simple and has been improved by many researchers. For example, Wang et al. [15] proposed combining PSSM with Conditional Neural Fields (CNF). The constructed DEEPCNF prediction algorithm has made a big break-through in the prediction of secondary structures of protein. In 2018, Ma and Liu et al. [16] proposed a method based on data partitioning and the semi-random subspace method (PSRSM). PSRSM is based on the traditional random subspace method, which was able to improve the accuracy of 85.89%, which effectively ensured the accuracy of the basic classifier. Yaseen et al. [17] studied a template-based method that began by constructing a structural template from a known protein structure with a certain sequence similarity and then incorporated the structural template as a feature with sequence and evolutionary information into two feedforward nerves. Prediction within the network, using 7-fold cross-validation, reached an accuracy of 80%. Heffernan et al. [18] used long short-term memory (LSTM) bidirectional recurrent neural networks (BRNNs) to capture the predicted protein residue types from the non-local interactions between amino acid sequence position distances. The team developed a tool, called SPIDER3 that achieved 84% accuracy of Q_3. In 2018, Fang et al. [19] considered local and global interactions between amino acids; they proposed a Deep3I deep neural network and developed the MUFOLD-SS tool, resulting in an accuracy of 85% for protein prediction. In [20], a prediction method, called DeepACLSTM, used a deep Asymmetric Convolutional Neural

Networks (ACNNs) and LSTM model to extract the ACNN, which is used to extract amino acids from local complex environments. LSTM is used to capture remote interdependencies. In [21], the authors also combined Convolutional Neural Networks and LSTM for prediction and achieved an accuracy of 70.7% for Q8 on CB513. Recently, [22, 23] proposed an end-to-end learning framework for protein contact prediction that can effectively use information obtained from deep or shallow MSAs, and the model is not only able to make accurate predictions of protein secondary structures, but also to improve the accuracy of predictions of 3D structures. Zhao et al. [24] used adversarial networks and convolutional neural network models to simulate strong correlations and long-range interactions between amino acid residues and achieved better prediction results.

The secondary structure of amino acid resides in a protein chain depends not only on the local structural information of the protein chain, it is also influenced by distant amino acid fragments. There is still no good method to effectively use these two aspects of information when predicting the secondary structure of a protein. Additionally, the difficulty of secondary structure prediction lies in the fact that the structures of different regions are too similar to extract effective information or suppress invalid information of protein sequences, making it difficult to discriminate between the structures of proteins. Thus, we propose a multi-scale multi-channel network model and correlation cross-entropy loss function (COCRS Loss) to solve the secondary structure prediction of protein. The main contributions of this research are as follows.

- This work designs a multi-scale, multi-channel convolutional network model, where each channel uses convolutional kernels of different scales to extract local feature information of different granularity. As the number of convolutional layers increases, the interaction information of the data at a distance is gradually extracted. This provides the model with the ability to simultaneously extract local features of protein sequences and capture remotely dependent information.
- This work uses a multi-channel model for multi-task feature extraction and fusion to achieve interoperability of feature information between multiple tasks, so that the model can more fully exploit the interaction information between protein sequences.
- This model introduces a convolutional attention mechanism between space and channels, and adaptively combines sequential relationships between information to reflect the importance of different positions. It enables the model to pay more attention to the important features in the sequence structure and to better obtain prediction accuracy.
- This work proposes correlation cross-entropy (COCRS Loss) as a loss function to automatically solve the unbalanced nature of the training samples. It improves the ability of the model to resist gradient dispersion, speeds up convergence of the model, and enhances the generalization ability of the model.

2. Related works

2.1. Dataset and coding

The publicly datasets used in this study include ASTRAL [25], CullPDB [26], Casp10 [27], Casp11 [28], Casp12 [29], and Casp9 [27]. We selected data based on a 25% percent identity cut, a 3 Å resolution cut, and an R-factor cut of 0.25. The specific experimental data are shown in Table 1. Protein secondary structure types were defined using the DSSP [30]. The DSSP has eight classes of

secondary structures: H (α-helix), G (3-helix), I (5-helix), E (fold), B (β-turn), T (turn), S (bend and roll), and helix (' _ '), which are usually divided into three classes. In the experiment, we substituted H, G, I for H; E, B for E; and other states for C, which usually results in lower prediction accuracy than other definitions. To encode the protein data, we used the PSI-BLAST [31] tool to call 3 iterations, with the detection evolution matrix set to a BLOSUM62 matrix and the E-value is set to 0.00l. Each protein sample in the dataset was searched to generate the corresponding PSSM matrix ($20 \times L$), where L is the length of the amino acid sequence, 20 represents the amino acid number of types, and each type represents the likelihood of a residue mutating into the corresponding amino acid type.

Table 1. Statistical data in datasets.

Datasets	Number of protein	Number of amino acids
Casp9	122	15193
Casp10	99	12742
Casp11	81	9040
Casp12	19	1717
ASTRALCULL	15696	3863251

To better obtain residue sequence information and predict the secondary structure of the central residue, we used a sliding window [24] to partition the PSSM matrix and set the sliding window value to 13 to transform the data. We are able to generate a feature code of 20×13, which was mainly used as input for the first channel in the model. Figure 1 shows the slicing of the amino acid sequence in one pass with a slicing window of size 13. To enrich the experiment, the sliding window values were again set to 19 and 27, at which point a protein feature was transformed into 20×19 and 20×27 feature codes, which were used as inputs for the second and third channels in the model.

M	D	P	F	L	V	L	L	H	S	V	S	S
-4	-2	1	-6	-4	0	-3	-5	-1	-1	-4	1	-1
-5	-4	0	-6	6	1	-1	-6	-2	2	-6	-4	-2
-6	0	-1	-6	2	-1	-4	-7	-1	1	-7	-1	1
-6	6	2	-7	-4	-1	-5	-7	-4	2	-7	3	2
-4	-5	-3	-6	-3	-2	2	-5	-2	-3	-4	-1	-4
-4	-3	-1	-6	2	2	0	-4	-2	1	-6	-2	1
-6	-1	0	-6	-3	0	-4	-7	-4	2	-6	-2	5
-6	-3	-2	-6	-4	-2	-5	-7	-4	-2	-7	2	-2
-5	-4	-3	-4	2	-3	-4	-6	5	0	-6	-3	-3
4	-6	-3	-4	-3	0	0	-1	-1	-2	7	-6	-5
0	-6	-3	-3	1	-1	5	6	3	-2	1	-6	-4
-5	-3	-1	-6	0	2	0	-6	-3	2	-6	-3	1
9	-5	-2	-3	-1	-2	4	4	1	-2	-1	-5	-2
-3	-6	-2	9	-3	-3	-1	-2	3	-2	2	-6	-5
-6	-4	6	-5	-5	-2	-5	-7	-5	-3	-6	-4	-4
-5	5	0	-5	-2	1	-3	-6	-2	2	-6	5	2
-4	-2	-1	-5	-4	0	1	-5	-1	-1	-4	-2	-2
-5	-7	-4	-3	-3	-4	-4	-5	-2	-5	-5	-6	-5
-4	-5	-2	5	-3	-2	-4	-5	3	-3	-4	-5	-4
3	-5	-3	-3	-1	2	0	-3	-2	-3	3	-5	-4

Figure 1. Part of the data after slicing and processing.

2.2. Convolutional neural network

This study uses convolutional neural networks (CNN) [32] to autono-mously extract amino acid residue features for adaptive learning. The neural network structure diagram is shown in Figure 2. It mainly learns to extract local features of the input data through convolutional and pooling layers to extract and retain important information of the representations.

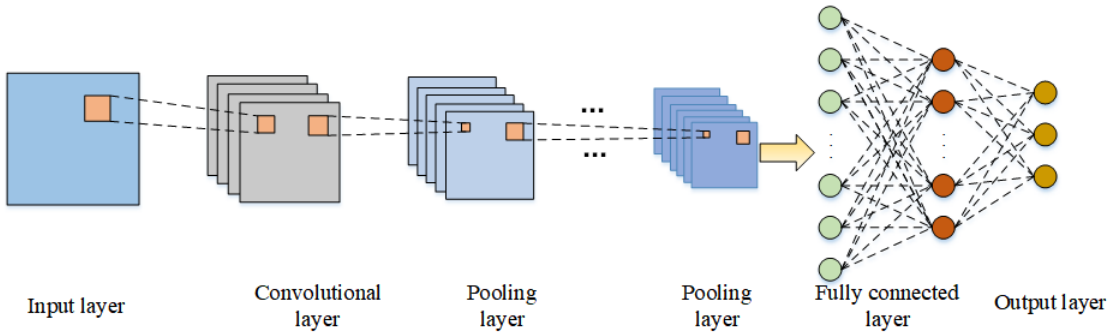


Figure 2. Convolutional neural network structure diagram.

Convolutional layers are typically used to process various forms of input data. Each convolutional layer consists of a Feature Map (FM) and a convolutional kernel, which moves in fixed steps over the Feature Map and performs a convolutional operation with the corresponding positions of the local field of perception, and finally passes through the activation function to obtain the output value. Which is defined as follows:

$$y_i^h = ReLU\left(\sum_{i=1}^{n} w_i^{h-1} \otimes x_i^h + b_i, 0\right) \tag{2.1}$$

where W^{h-1} is the region map generated from the input amino acid PSSM encoding matrix and the convolution kernel of the previous layer. y_i is the feature map obtained after convolution of the $h-th$ layer. b_i is the corresponding weight bias, and $ReLU$ is a nonlinear activation function [33]. Usually in CNN, each layer of feature data is subjected to a non-linear activation function to activate the data. In this study, we use the sigmoid and $ReLU$ activation functions in Figure 3 to activate the weighted data. In this study, we use the sigmoid and $ReLU$ activation functions in Figure 3 to activate the weighted data. It can be seen that the softsign function has a smoother curve and slower rate of derivative decline than the $ReLU$ function, which to some extent alleviates the problem of slow convergence and gradient disappearance.

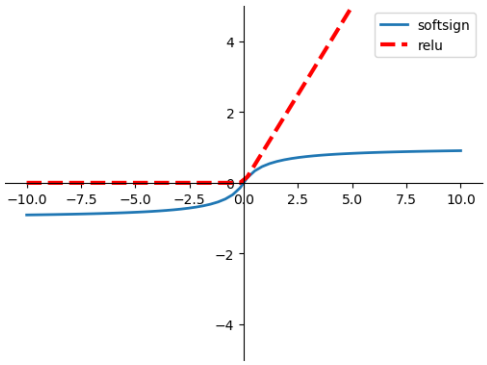


Figure 3. Schematic representation of the *ReLU* and softsign activation functions.

The pooling layer acts as a secondary extraction of features, extracting the most important feature information in the sequence on the basis of reducing the number of feature values. The fully connected layer is the process of linearly transforming the extracted features from one feature space to another, and its core operation is the matrix vector product. In this paper, the Dropout mechanism is used for the fully-connected layer. By randomly "pausing" and "restarting" the neurons in the convolutional neural network with a certain probability, the network's ability to fit the data is effectively improved. The Dropout expression of the fully connected layer is shown in Eq (2.2).

$$y_i = \mathrm{w}_i \left(x_i \otimes b_i \right) \tag{2.2}$$

where x_i represents the input vector of the $i - th$ layer, y_i represents the corresponding output vector, w_i represents the weight matrix of the $i - th$ layer, and b_i is a mask that obeys the binomial bernoulli distribution. In the training phase, the dropout method randomly sets the mask to be 0 with a certain probability. Finally, the network outputs the probability that each sample belongs to the corresponding class through the softmax classifier.

2.3. Attention module

In order to improve the ability of the model to extract key features. We add Convolutional Block Attention Module (CBAM) [34] to the CNN to extract the key features. The CBAM structure is shown in Figure 4. The input feature map is sequentially obtained through the channel and spatial attention module to obtain the refined feature map (Refined Feature), which can be seen as a significant feature extracted in both the channel and spatial dimensions.

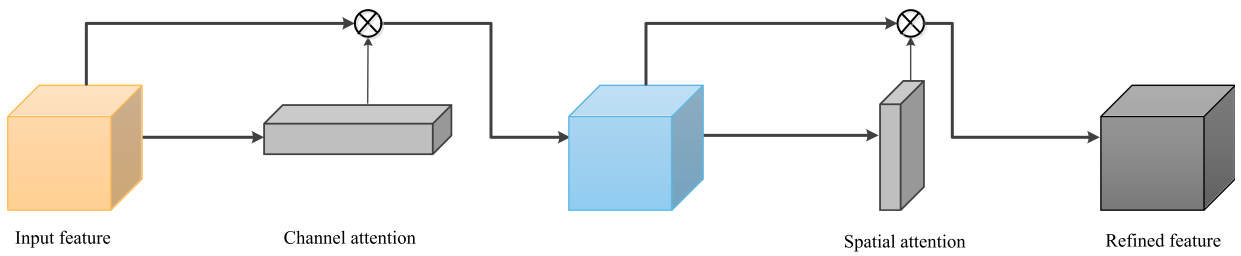
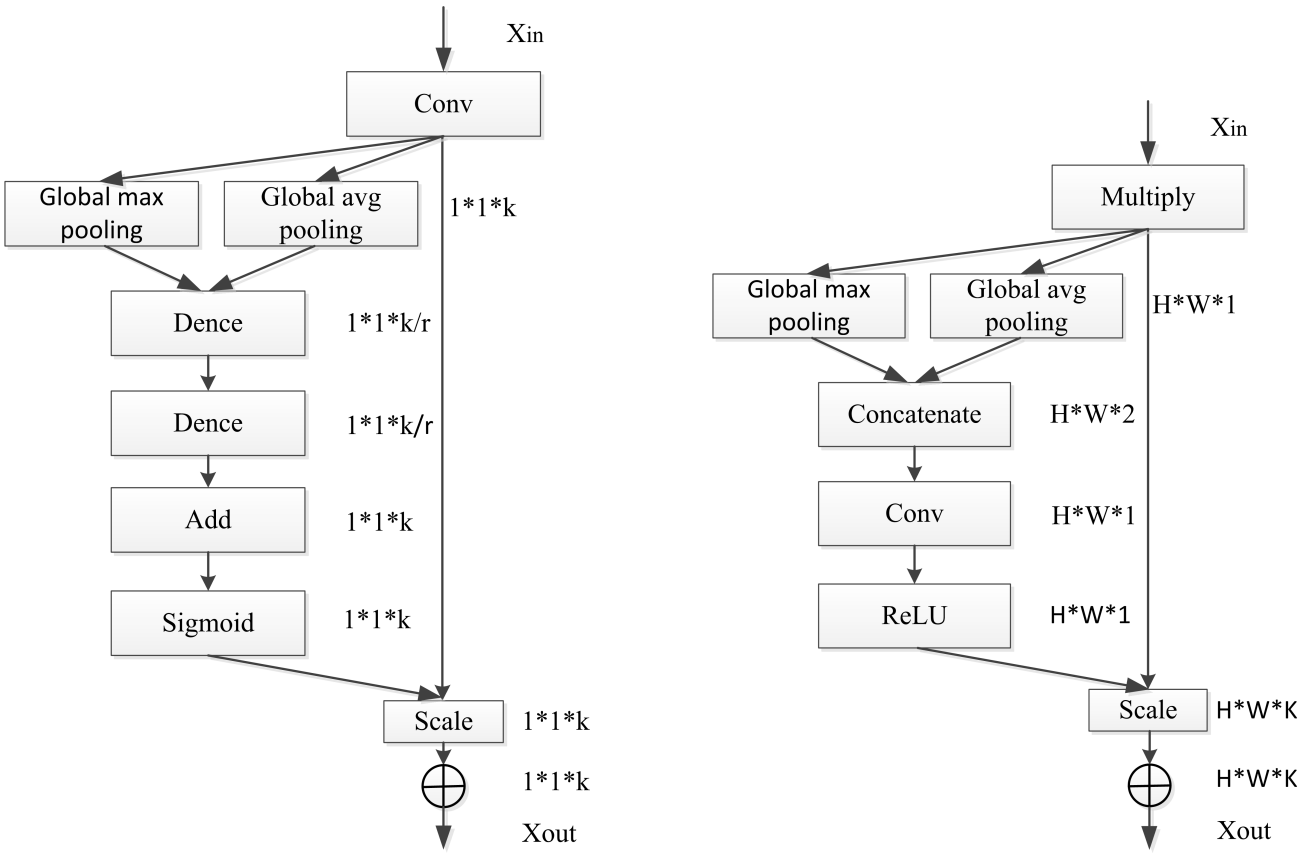


Figure 4. Convolutional Block Attention Module (CBAM) flowchart.



(a) Diagram of the Channel attention module process.

(b) Diagram of the spatial attention module process.

Figure 5. Diagram attention module process.

2.3.1. Channel attention block

In order to obtain the channel features, the convolutional feature matrix $x_k(i,\ j) \in R^{H\times W\times K}$ is used as input (H, W are the length and width of the feature map, K represents the number of filters). We provide a diagram in Figure 5(a) to depict the detailed process of the channel attention module. The input x_k is subjected to a basic convolution operation and the resulting features then enter the global average pooling and global maximum pooling branches for the purpose of feature enrichment. The global average pooling calculation is shown in Eq (2.3):

$$x_k^a = ReLU\left(\frac{1}{H\times W}\sum_{i=1}^{H}\sum_{j=1}^{W} x_k(i,j), 0\right) \tag{2.3}$$

where x_k^a is the average pooling weight of the $k-th$ filter of the output. $x_k(i,\ j)$ is the input feature layer. $H \times W$ represents information about the spatial dimension of the feature matrix. $ReLU()$ is the activation function. The global maximum pooling calculation is shown in Eq (2.4):

$$x_k^m = \max\left[ReLU\left(x_k(i,j)\right), 0\right] \quad \mathrm{i} = 1\ldots H \quad j = 1\ldots W \tag{2.4}$$

where x_k^m is the maximum pooling weight of the $k-th$ filter of the output. $x_k(i,\ j)$ is the input feature layer. $ReLU()$ is the activation function.

The output two feature maps are then summed element by element and fed into a fully connected layer with compression operation (r is the compression ratio), and activated using the Sigmoid function to obtain weight coefficients of size $1 \times 1 \times k$. Here k represents the number of filters. Finally, the original features are multiplied by the obtained weighting factor to obtain the inter-channel attention features.

2.3.2. Spatial attention block

In order to generate corresponding spatial attention weights to reflect the characteristics of different spatial locations, we take the channel attention feature map $x_k^{chan} \in R^{H\times W\times K}$as the input ($H$, W is the length and width of the feature map, and K is the number of channels). We provide a diagram in Figure 5(b) to depict the detailed process of the spatial attention module. Similar to our channel attention module, we perform the maximum pooling and average pooling operations along the channel dimension. After pooling, two feature descriptions of size $W \times H \times 1$ and $F_k^a \in R^{1\times 1\times K}$ and $F_k^m \in R^{1\times 1\times K}$ are obtained, and the two feature descriptions are stitched together in a cascading manner to maintain the diversity of network information.Then it flows into the convolutional layer with shrinkage operation. After the shrinkage operation, a nonlinear transformation is performed using the *ReLU* function. The weight coefficients of size $H \times W \times 1$ are obtained. Finally, we multiply the original feature and the weight coefficient to get the enhanced feature map. The procedure for calculating maximum and average pooling in the spatial attention module is shown in Eqs (2.5) and (2.6).

$$F_s^a = ReLU\left(\frac{1}{k}\sum_{i=1}^{k} z_s(i), 0\right) \tag{2.5}$$

where F_s^a represents the spatial information weight generated at the $s-th$ position, and z_s represents the form of converting x_k^{chan} into a vector, where $s = H \times W$. $ReLU()$ is the activation function.

$$F_s^{\mathrm{m}} = \max\left[ReLU\left(\mathrm{z}_s(\mathrm{i})\right), 0\right] \quad \mathrm{i} = 1\cdots\mathrm{k} \tag{2.6}$$

where F_s^m is the spatial maximum pooling weight generated at the $s-th$ position of the output.

3. Proposed approach

Single-channel neural networks, with a single convolutional kernel size, leads to the inability to capture long-term dependencies between the same protein sequences. And multiple channels will lead to an increase in parameters, making the model severely over-fitted, which in turn reduces the classification accuracy of the data. To address the above problems, this study proposes to propose a multi-scale convolutional and correlated cross-entropy loss function model (COCRS Loss) incorporating a convolutional attention mechanism to solve the problems in protein secondary structure prediction. The network model is divided into four main parts, amino acid matrix encoding, multi-scale parallel convolutional block extraction, feature fusion, and prediction classification. The model flow chart is shown in Figure 6.

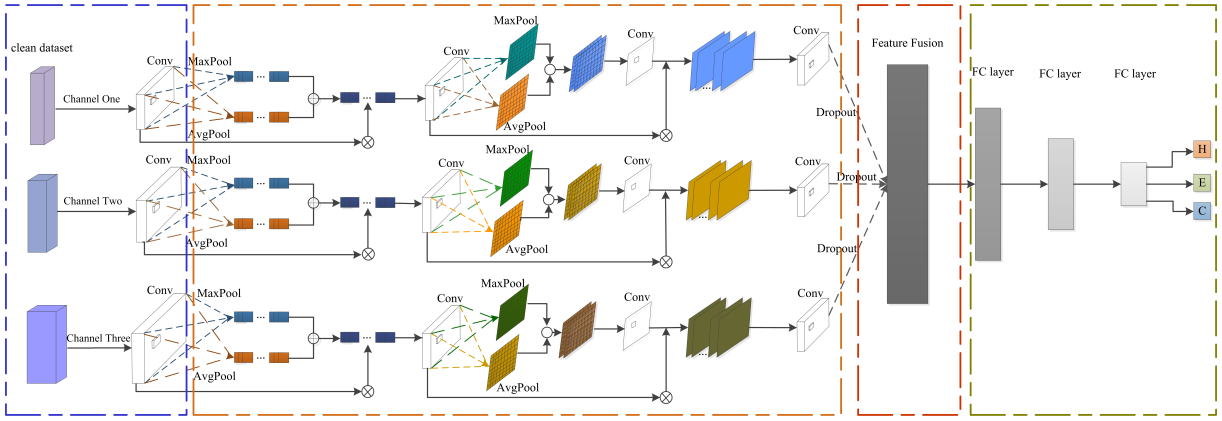


Figure 6. The main flow chart of the multi scale convolutional attention neural network model.

3.1. Amino acid matrix coding

We use the window parameters [24] to preprocess protein sequence information to obtain residue sequence information and predict the secondary structure of central residues. Taking the middle amino acid as the reference, the middle position of the window was overlapped with the first valid amino acid character by using 13, 19, 27 as the window size and moving one position towards the end of the amino acid sequence in turn until the number of moves equal to the total length of the amino acid sequence of the current slicing window. The current amino acid sequence window slicing processing was completed. The processing resulted in $P_a(i)^{20\times13}$ $i = 1....N$,$P_b(i)^{20\times19}$ $i = 1....N$,$P_c(i)^{20\times27}$ $i = 1....N$, three sets of input data, where N is the total number of samples. Figure 1 shows the pre-processing results when the window slice is 13.

3.2. Multi-scale parallel convolutional block extraction

For the feature extraction module, we used a multi scale parallel architecture to extract feature information from the amino acid samples. Table 2 shows the model's each channel parameter changes. Where the convolution motion step size is 1, and the output size respectively represents the length, width and number of channels of the data. Taking the first channel as an example, we first set the convolution kernel $C1 \times C1$ of the same size, and perform preliminary feature extraction on the input data via the convolution layer to obtain the feature map $conv1_C$. In order to make the model better perceive the key information in the feature map, we perform two attention perceptions on the feature map. We make the feature map $conv1_C$ perform the global maximum pooling and global average pooling operations respectively to obtain two feature vectors $conv1_C^{avg}$ and $conv1_C^{max}$, input them into the same multi-layer perceptron and add the results element by element to get a $1 \times 1 \times 430$ feature weight vector. Next, we multiply the feature weight vector and $conv1_C$, element by element, to get the attention feature matrix M_C^{chan} between channels. The equation is as follows:

$$\mathrm{M}_c^{\mathrm{chan}} = \sigma\left(MLP\left(\mathrm{conv}\ 1_c^{\mathrm{avg}}\right) \oplus MLP\left(\mathrm{conv}\ 1_c^{\mathrm{max}}\right)\right) \otimes \mathrm{conv}\ 1_c \tag{3.1}$$

where $\oplus$ represents element by element addition, $\otimes$ represents element by element multiplication, and $\sigma()$ represents the Sigmoid activation function. The calculation process of $conv1_C^{avg}$ and $conv1_C^{max}$

eigenvectors is shown in Eqs (2.3) and (2.4). On the basis of the feature vector M_C^{chan}, the global maximum pooling and global average pooling operations are performed again along the channel space dimension, and the feature vectors N_S^{avg} and N_S^{max} are obtained. Connect the output results in parallel, and after convolution operation, a feature weight vector of $20 \times 13 \times 1$ is obtained, and the feature weight vector is multiplied by M_C^{chan} element by element to obtain the reconstructed feature matrix N_C^{spat}.The calculation equation is as follows:

$$\mathrm{N}_c^{\mathrm{spat}} = \sigma\left(\mathrm{cov}\left(\mathrm{N}_S^{\mathrm{avg}} \circledast \mathrm{N}_S^{\mathrm{max}}\right)\right) \otimes \mathrm{M}_C^{\mathrm{chan}} \tag{3.2}$$

where $\circledast$ represents the feature element parallel operation, $cov()$ represents the convolution operation on the merged pooling matrix, and $\sigma()$ represents the Sigmoid activation function. The reconstructed feature matrix N_c^{spat} is then subjected to another convolution, a dropout operation, which excites the information of local regions in the lower levels of the reconstructed features to higher levels via the convolution kernel. This highlights the important features in the data. For the other channels, convolutional kernels $C2 \times C2$, and $C3 \times C3$ of different sizes from the first channel are set to perform convolutional operations on the other two amino acid codes in order to extract local features different from those of the first channel. We find that the network structure proposed in this paper extracts features of different granularity (e.g., input data of different encodings) during the training of multiple channel tasks, thus achieving improved performance for each channel task. Compared with the single-channel task model that directly uses one encoding, the multi-channel task model with multiple encodings has the advantages of greater generalization of features extracted from the data, allowing each channel task to have independent feature mechanisms and more fully extracting local feature information from the three encodings.

Table 2. The parameter settings of each channel in the proposed model.

	Layer name	Kernel size	Output size ($c \times h \times w$)
Channel 1	input	–	$20 \times 13 \times 1$
	conv1	12×12	$20 \times 13 \times 430$
	attention	–	$20 \times 13 \times 430$
	conv2	3×3	$18 \times 11 \times 530$
	dropout	–	$18 \times 11 \times 530$
Channel 2	input	–	$20 \times 19 \times 1$
	conv1	18×18	$20 \times 19 \times 430$
	attention	–	$20 \times 19 \times 430$
	conv2	6×6	$15 \times 14 \times 530$
	dropout	–	$15 \times 14 \times 530$
Channel 3	input	–	$20 \times 27 \times 1$
	conv1	26×26	$20 \times 27 \times 430$
	attention	–	$20 \times 27 \times 430$
	conv2	9×9	$12 \times 19 \times 530$
	dropout	–	$12 \times 19 \times 530$

3.3. Feature fusion

We fused the extracted feature data of each channel after re-convolution, and the feature fusion process is shown in Figure 7. Since the 9 × 9 convolutional kernels of channel 3 are stacked, the sensory field is greater than the 3 × 3 convolutional kernels of channel 1 and the 6 × 6 convolutional kernels of channel 2. Thus, the network structure is deeper and has a larger field of view. Therefore, the feature fusion part can make better use of the coding features extracted from each channel and provide more useful training data for the prediction phase.

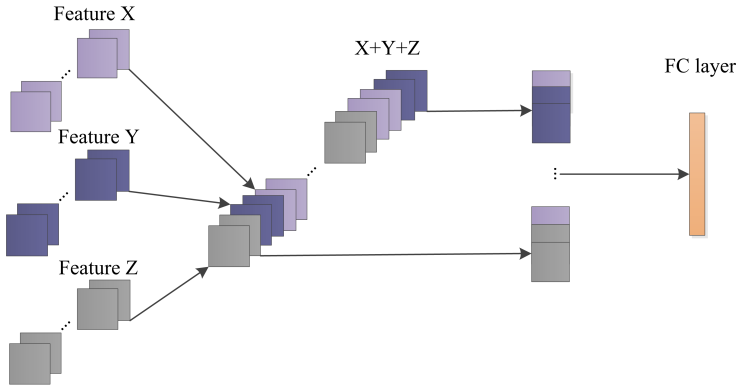


Figure 7. The detail diagram of the feature fusion part of the proposed model.

3.4. Prediction classification

In the classification prediction part, the model operates with a 3-layer fully connected layer, where the input layer receives the fused feature vectors, and the output layer uses a Softmax classifier to predict the accuracy of the secondary structure. While the traditional cross-entropy loss function only considers the separability of features during the model training process, it does not consider the training objective of intra- and inter-class similarity of vectors. In this study, a correlation measure is added based on the cross-entropy loss function, and the covariance and standard deviation are used to calculate the positive and negative correlations between the predicted vectors and the real vectors within and between classes. Taking any training sample as an example, it is assumed that the probability of an amino acid type output by the Softmax function is $\hat{y} = [\hat{y}_1, \hat{y}_2 \ldots \hat{y}_C]$, and the true label is $y = [y_1, y_2 \ldots y_C]$ (in one-hot encoding form), C is the total class of the sample. The COCRS loss function is expressed as:

$$\rho_{loss} = -\left[\sum_{i=1}^{C} y_i * \log\left(\hat{y}_i\right)\right] * \left\{1 - \left[\sum_{i=1}^{C} \left(y_i - \bar{y}\right)\left(\hat{y}_i - \hat{y}_i\right) / \sigma(y)\sigma(\hat{y})\right]^2\right\} \tag{3.3}$$

where $\bar{y}$,$\bar{\hat{y}}_i$ are the means of the vectors y and $\widehat{y}$, and $\sigma(y)$and $\sigma(y)$ are the standard deviations of y and . Let $y'_i = y_i - \bar{y}, \widehat{y'_i} = \widehat{y_i} - \bar{\hat{y}}_i$, Eq (10) can be reduced to:

$$\rho_{loss} = -\left[\sum_{i=1}^{C} y_i * \log\left(\widehat{y_i}\right)\right] * \left\{1 - \left[\frac{\sum_{i=1}^{C} y'_i \hat{y}'_i}{\sqrt{\sum_{i=1}^{C} y'^2_i}\sqrt{\sum_{i=1}^{C} \hat{y}'^2_i}}\right]^2\right\} \tag{3.4}$$

As can be seen from the formula, the correlation measure is obtained by dividing the covariance by the standard deviation of the two variables, and the magnitude of the value can be a good measure of the degree of association between the two random variables. In Eq (3.4), the cross-entropy loss function maximizes the output probability of the true classes to which they belong, thus making the features of different classes differentiable. The correlation measure term, on the other hand, better presents the negative correlation between the predicted samples and the samples of different classes by means of linear regression, while better presenting the positive correlation with the samples of the same class. This allows for greater inter-class distances and smaller intra-class distances for amino acid features, enhancing the learning ability of the model.

4. Experiment and analysis

This study sets up a series of comprehensive experiments to evaluate the effectiveness of the proposed depth model. First, the effect of encoding the data at different scales on the accuracy was recorded. This is followed by an ablation study within the model to explain the contribution of each component of the model to the final results. Finally, the overall performance of the network model on Casp9 [27], Casp10 [27], Casp11 [28] and Casp12 [29] is presented and compared fairly with other methods.

4.1. Evaluation index

In this paper, accuracy Q_3 and the segment overlap measure (Sov) [35] are used to measure the performance of protein secondary structure prediction. Q_3 mainly measures the accuracy of individual residue allocation, the expression is:

$$Q_3 = \frac{N_C + N_E + N_H}{N} \times 100 \tag{4.1}$$

where N represents the total number of amino acid residues, and N_C, N_E, and N_H are the predicted correct numbers in the spiral, fold, and random coil, respectively. The accuracy of any class of these secondary structures can be equation as follows:

$$Q_c = \frac{TP_c}{n_c} \times 100 \tag{4.2}$$

where TP_c is the correct predicted number of amino acid residues in Class C, and n_c represents the total number of amino acid residues in Class C in the data. Segment Overlap (Sov) a measure based on secondary structure fragmentation, the Sov is calculated as shown in Eq (4.3).

$$Sov_3(\%) = \frac{1}{m}\left(\sum_{i\in H,E,L\cdots}\sum_{S(i)}\left[\frac{\min OV(s_1,s_2) + \delta(s_1,s_2)}{\max OV(s_1,s_2)} \times \text{len}(s_1)\right]\right) \times 100 \tag{4.3}$$

s_1, and s_2, in Eq (4.3) refer to the actual and predicted secondary structures that are fragments of a certain class, respectively, $S_{(i)}$ is the number of all fragment pairs (s_1, s_2), $\min OV(s_1, s_2)$ is the actual overlap length of s_1, and s_2, and $\max OV(s_1, s_2)$ is the greater of the length of either s_1, or s_2, that contains the entire fragment corresponding to the structural residues of a certain class m denotes the total number of amino acid residues. For $\delta(s_1, s_2)$ the formula is shown in Eq (4.4).

$$\delta(s_1, s_2) = \min \left\{ \begin{array}{c} \max OV(s_1, s_2) - \min OV(s_1, s_2) \\ \min OV(s_1, s_2) \\ \text{int}(0.5 \times \text{len}(s_1)) \\ \text{int}(0.5 \times \text{len}(s_2)) \end{array} \right\} \tag{4.4}$$

where len (s_1) and len (s_2) refer to the total number of amino acid residues contained in s_1 and s_2.

4.2. Impact of data coding on accuracy

In this study, we apply datasets ASTRAL+CullDB as the training set and Casp9, Casp10, Casp11, and Casp12 as the test sets for experiments. Because all experiments use the PSSM matrix as the input of the model, the length of the sliding window determines the prediction effect when the PSSM encoding of the amino acid data is sliding. In order to verify the effect of different windows on the data, the protein data under 13 and 19 and 27 windows were validated on Casp10 in this paper, and the experimental results are shown in Table 3.

Table 3. The effect of different window sizes on the data (%).

Window sizes	Q_3	Q_H	Q_E	Q_C
13	84.61	87.69	78.87	84.79
19	86.91	90.49	82.17	86.84
27	88.23	90.52	85.96	87.76

It can be seen that the maximum accuracy of PSSM coding is 88.23% when the sliding window is 27. Secondly, when the sliding window is 19 and 13, the accuracy results are better, 86.91 and 84.61% respectively. Experiments show that the larger the passage window, the larger the amount of protein data selected will have a greater impact on the prediction results.

4.3. Model overall analysis

To better understand the proposed depth model, two empirical studies were conducted. In the first stage, a 10-fold cross-validation test method is used to test the model. In the second stage, we did an exploratory comparison of each of these components. First, set the hyperparameters of the whole model. The size of the convolutional kernel selected for each channel is shown in Table 2. In the training process, the batch size is 128, the convolutional step size is 1, and the keep prob index of dropout is 0.5. For the 3 fully connected layers, the number of hidden units is 128, 64, and 3, respectively. In addition, the model uses the inverse gradient training descent algorithm Adam to train the weight parameters of the network, where the momentum is set to 0.9, the weight decay is set to 10^{-4}. Then we uses the 10-fold cross-validation test method to test the accuracy of the model. Set K=10, i.e., divide the ASTRAL+CullDB dataset into 10 equal parts, and take turns using 9 parts of them as training data and 1 part as test data for testing. After one round of verification, the test accuracy is 89.99%, 89.32%, 88.93%, 88.45%, 89.14%, 89.57%, 89.71%, 88.64%, 89.68% and 88.25% respectively, and the average classification accuracy is 89.168%.

Second, we performed ablation studies on the dataset Casp10 by deleting or replacing individual modules in the present study model. We performed multi-scale single-layer convolution

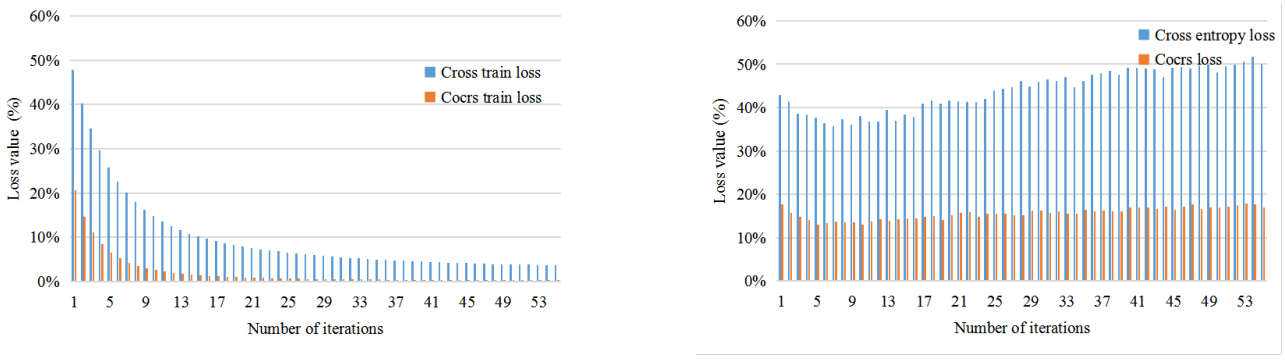
(MSSL_COV), multi-scale single-layer convolution with CBAM (MSSL_COVATT), multi-scale multi-layer convolution with CBAM (MSML_SOVATT), and replacement of conventional cross-entropy with correlated cross-entropy loss experiments (This work model), respectively.

Table 4. Comparison of prediction performance of each model in ablation experiment.

Methods	Q_H	Q_E	Q_C	Q_3	Sov
MSSL_COV	0.8971	0.8556	0.8678	0.8746	0.7887
MSSL_COVATT	0.8942	0.8205	0.8648	0.8637	0.7770
MSML_SOVATT	0.9212	0.8861	0.8915	0.9000	0.8500
This work model	0.9167	0.8978	0.8898	0.9007	0.8547

From the results of the various ablation experiments in Table 4, we can see that multi-scale and multi-convolution have an important impact on the network model. The prediction accuracy Q_3 of the MSML_SOVATT model reaches 90.00%, which is about 2.5% higher than that of the MSSL_COV model. The Sov accuracy is 85%, which is about 7% higher than the Sov rate of the MSSL_COV model. After replacing the traditional cross entropy with our proposed COCRS loss function, the prediction result reaches 90.01%, which is 0.07% higher than the prediction accuracy using the traditional cross entropy loss, while the Sov accuracy reaches 85.47%. Thus, the network model proposed in this study with the extraction of feature information and fusion of local feature information by picking multi-scale convolution and convolutional attention blocks, and training with the COCRS loss function is effective.

The experiments also documented the comparison of model training on the Casp10 test set using both loss functions. Figure 8(a) shows the convergence of the training loss in the two loss models and Figure 8(b) shows the convergence of the test loss in the two loss models.



(a) The convergence of the training loss in the model.

(b) The convergence of the test loss in the model.

Figure 8. The comparison between traditional cross entropy loss and CROCS loss in the model test loss.

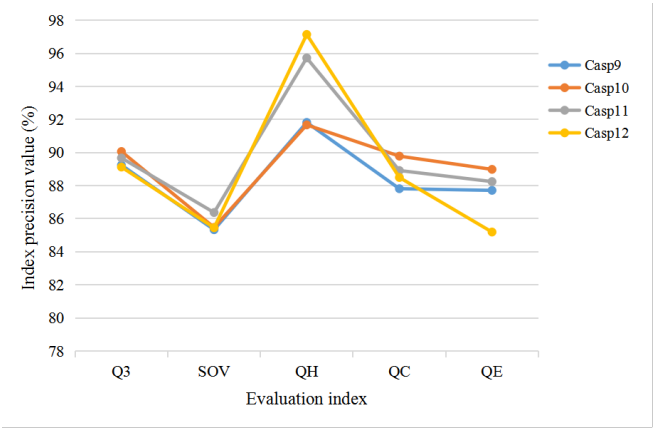


Figure 9. Evaluation accuracy of different data sets.

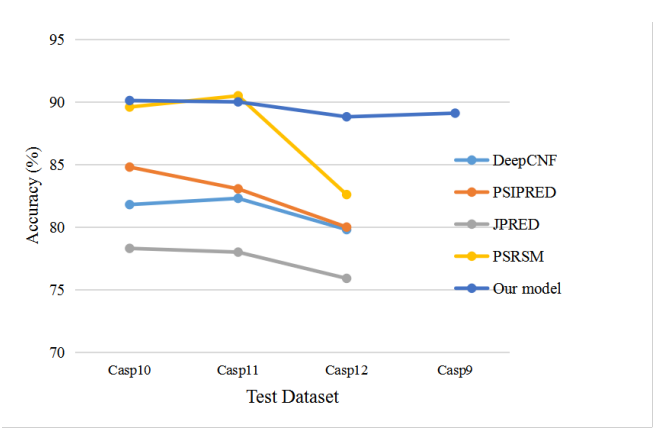


Figure 10. The Q3 accuracy of the comparison method on the four data sets.

Figure 8 show that the initial loss value of the COCRS loss model is 0.2055 at the beginning of the iteration, and the model Loss value decreases continuously with the increase of the number of iterations. In the 20th iteration of the training phase, the loss value of this model decreases to 0.0096 and 0.1415 on the training and test sets, respectively, while the loss value of the cross-entropy loss model is 0.0821 and 0.4089 on the training and test sets, respectively. The convergence speed of the traditional cross-entropy loss function is faster. In the middle of the iteration, the COCRS function fluctuates more slowly due to the correlation coefficient term, while the traditional cross-entropy loss function fluctuates more. In the late iteration, both the COCRS loss function and the traditional cross-entropy loss function converge relatively well, but the misclassification costs of the COCRS loss function is maintained at a relatively low level in the late iteration, and the misclassification cost at the final convergence is substantially lower compared with the traditional cross-entropy loss function. In general, the convergence speed of the COCRS loss function is better than that of the traditional cross-entropy loss function, and the final misclassification cost is substantially lower than that of the traditional cross-entropy loss function. In addition, we conducted overall experiments on the Casp9, Casp10, Casp11, Casp12 dataset separately to demonstrate the effectiveness of the proposed model. The evaluation results are shown in Figure 9.

4.4. Compare with other models

Table 5. Comparison of prediction accuracy Q_3 (%) of each model on the four datasets.

Methods	Casp10	Casp11	Casp12	Casp9
DeepCNF	81.81	82.33	79.80	–
PSIPRED	84.83	83.06	80.04	–
JPRED	78.30	78.02	75.92	–
PSRSM	89.61	90.53	82.62	–
This work model	90.12	90.01	88.82	89.11

To further validate the prediction performance of the model in this study, we compare the present model with other models, in Table 5 and Figure 10, the comparison models are DeepCNF [15], PSRSM [16], PSIPRED [14], and JPRED [36]. It can be seen from Table 5, in the CASP10 test set, the test results of this model were higher than those of DeepCNF, PSRSM, PSIPRED, JPRED methods, with an accuracy of 90.12%. It proved that the model could improve the accuracy of amino acid prediction. The prediction results of CASP11 remained almost the same as those of PSRSM model, but both were higher than those of DeepCNF, PSIPRED, and JPRED. The accuracy of CASP12 dataset was improved by 5.6% compared with PSRSM method. It improved by 8.8% over the PSIPRED method. The prediction accuracy of this research model is significantly higher than other comparison models. It can be seen that this model fully extracts the type information coding of amino acids and biological evolution structure information, and effectively interacts the extracted local features to improve the accuracy of the protein secondary structure.

5. Conclusions and future work

In this study, a multi-scale convolutional attention neural network model is proposed for the problems that the prominent position information of protein sequences cannot be extracted well and the structure of the protein is difficult to distinguish under the interference of invalid information. The model uses amino acid codes of different scales (13, 19, and 27) to be input to multiple channel tasks for training. Compared with the previous research, the model has stronger generalization ability of features extracted from the data, which makes each channel task have an independent feature mechanism, and fully extracts the local feature information in the three kinds of codes. Considering that the traditional cross-entropy loss does not effectively solve the problem of sample non-equilibrium in the training sequence, we propose an improved correlation cross-entropy function (COCRS Loss) as the loss function to self-adapt to solve the non-equilibrium of training samples based on the present model.

A large number of experiments conducted on the public data sets Casp9, Casp10, Casp11 and Casp12 of this model show that our model can better extract amino acid feature information. The improvement of the loss function speeds up the model convergence speed, improves the learning and generalization ability of the model, and prevents over-fitting at the same time, thereby achieving better prediction results. In the follow-up research work, we consider replacing the ordinary convolutional layer by deep separable convolution to further reduce the overhead of the network module.

Acknowledgments

This work is supported by Key Research and Development Project of Shandong Province (2019JZZY020124), China, and Natural Science Foundation of Shandong Province (23170807), China.

Conflict of interest

The authors declare that there is no conflict of interests regarding the publication of this article.

References

1. S. M. Najibi, M. Maadooliat, L. Zhou, J. Z. Huang, X. Gao, Protein structure classification and loop modeling using multiple ramachandran distributions, *Comput. Struct. Biotechnol. J.*, **15** (2017), 243–254.
2. J. Peng, D. Schwartz, J. E. Elias, C. C. Thoreen, D. Cheng, G. Marsischky, et al., A proteomics approach to understanding protein ubiquitination, *Nat. Biotechnol.*, **21** (2003), 921–926.
3. R. A. Cairns, I. S. Harris, T. W. Mak, Regulation of cancer cell metabolism, *Nat. Rev. Cancer*, **11** (2011), 85–95.
4. X. M. Zhao, R. S. Wang, L. Chen, A. Kazuyuki, Uncovering signal transduction networks from high-throughput data by integer linear programming, *Nucleic Acids Res.*, **36** (2008), e48.
5. P. Y. Chou, G. D. Fasman, Prediction of protein conformation, *Biochemistry*, **13** (1974), 222–245.
6. A. Anand, G. Pugalenthi, P. N. Suganthan, Predicting protein structural class by svm with class-wise optimized features and decision probabilities, *J. Theor. Biol.*, **253** (2008), 375–380.
7. Z. Tang, T. Li, R. Liu, W. Xiong, G. Chen, Improving the performance of β-turn prediction using predicted shape strings and a two-layer support vector machine model, *BMC Bioinf.*, **12** (2011), 283.
8. S. A. Malekpour, S. Naghizadeh, H. Pezeshk, M. Sadeghi, C. Eslahchi, Protein secondary structure prediction using three neural networks and a segmental semi markov model, *Math. Biosci.*, **217** (2009), 145–150.
9. I. Kurniawan, T. Haryanto, L. S. Hasibuan, M. A. Agmalaro, Combining pssm and physicochemical feature for protein structure prediction with support vector machine, *J. Phys. Confer.*, **835** (2017), 012006.
10. Y. Chen, J. Cheng, Y. Liu, P. S. Park, A novel approach of protein secondary structure prediction by svm using pssm combined by sequence features, in *Proceedings of SAI Intelligent Systems Conference*, Springer, Cham, 2016.
11. Y. Liu, J. Cheng, Y. Ma, Y. Chen, Protein secondary structure prediction based on two dimensional deep convolutional neural networks, in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, 2017.
12. M. Alirezaee, Ensemble of neural networks to solve class imbalance problem of protein secondary structure prediction, *Int. J. Artif. Intell. Appl.*, **3** (2012), 9–20.
13. B. Zhang, J. Li, L. Qiang, Prediction of 8-state protein secondary structures by a novel deep learning architecture, *BMC Bioinf.*, **19** (2018), 293.
14. L. J. Mcguffin, K. Bryson, D. T. Jones, The psipred protein structure prediction server, *Bioinformatics*, **16** (2000), 404–405.
15. Z. Wang, F. Zhao, P. Jian, J. Xu, Protein 8-class secondary structure prediction using conditional neural fields, *Proteomics*, **11** (2011), 3786–3792.
16. Y. Ma, Y. Liu, J. Cheng, Protein secondary structure prediction based on data partition and semi-random subspace method, *Sci. Rep.*, **8** (2018), 1–10.

17. A. Yaseen, Y. Li, Template-based c8-scorpion: a protein 8-state secondary structure prediction method using structural information and context-based features, *BMC Bioinf.*, **15** (2014), 1–8.
18. R. Heffernan, Y. Yang, K. Paliwal, Y. Zhou, Capturing non-local interactions by long short term memory bidirectional recurrent neural networks for improving prediction of protein secondary structure, backbone angles, contact numbers, and solvent accessibility, *Bioinformatics*, **33** (2017), 2842–2849
19. C. Fang, Y. Shang, X. Dong, Mufold-ss: New deep inception-inside-inception networks for protein secondary structure prediction, *Proteins: Struct., Funct., Genet.*, **86** (2018), 592–598.
20. Y. Guo, W. Li, B. Wang, H. Liu, D. Zhou, Deepaclstm: deep asymmetric convolutional long short-term memory neural models for protein secondary structure prediction, *BMC Bioinf.*, **20** (2019), 1–12.
21. I. Drori, I. Dwivedi, P. Shrestha, J. Wan, Y. Wang, Y. He, et al., High quality prediction of protein q8 secondary structure by diverse neural network architectures, in *32nd Conference on Neural Information Processing Systems*, Montreal, Canada, 2018.
22. C. Fang, Z. Li, D. Xu, Y. Shang, Mufold-ssw: A new web server for predicting protein secondary structures, torsion angles, and turns, *Bioinformatics*, **36** (2020), 1293–1295.
23. G. Xu, Q. Wang, J. Ma, Opus-tass: A protein backbone torsion angles and secondary structure predictor based on ensemble neural networks, *Bioinformatics*, **36** (2020), 5021–5026.
24. Y. Zhao, H. Zhang, Y. Liu, Protein secondary structure prediction based on generative confrontation and convolutional neural network, *IEEE Access*, **8** (2020), 199171–199178.
25. W. Kabsch, C. Sander, Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features, *Biopolym.: Orig. Res. Biomol.*, **22** (1983), 2577–2637.
26. G. Wang, R. L. Dunbrack, Pisces: recent improvements to a pdb sequence culling server, *Nucleic Acids Res.*, **33** (2005), W94–W98.
27. J. Moult, K. Fidelis, A. Kryshtafovych, A. Tramontano, Critical assessment of methods of protein structure prediction (CASP)-round IX, *Proteins: Struct., Funct., Bioinf.*, **79** (2011), 1–5.
28. J. Moult, K. Fidelis, A. Kryshtafovych, T. Schwede, A. Tramontano, Critical assessment of methods of protein structure prediction (CASP)-round X, *Proteins: Struct., Funct., Bioinf.*, **82** (2013), 3–9.
29. J. Moult, K. Fidelis, A. Kryshtafovych, T. Schwede, A. Tramontano, Critical assessment of methods of protein structure prediction (CASP)-round XII, *Proteins: Struct., Funct., Bioinf.*, **86** (2017), 7–15.
30. W. Kabsch, C. Sander, Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features, *Biopolymers*, **57** (2010), 75–80.
31. D. T. Jones, Protein secondary structure prediction based on position-specific scoring matrices., *J. Molecul. Biol.*, **292** (1999), 195–202.
32. G. Hu, X. Yang, Y. Zhang, M. Wan, Identification of tea leaf diseases by using an improved deep convolutional neural network, *Sustainable Comput.: Infor. Syst.*, **24** (2019), 100353.
33. N. Abramson, D. J. Braverman, G. S. Sebestyen, Pattern recognition and machine learning, *Publ. Am. Stat. Assoc.*, **103** (2006), 886–887.

34. S. Woo, J. Park, J. Y. Lee, I. S. Kweon, Cbam: Convolutional block attention module, in *Computer Vision – ECCV 2018. Lecture Notes in Computer Science*, Springer, Cham, 2018.

35. S. Wang, J. Peng, J. Ma, J. Xu, Protein secondary structure prediction using deep convolutional neural fields, *Sci. Rep.*, **6** (2016), 886–887.

36. A. Drozdetskiy, C. Cole, J. Procter, G. J. Barton, JPred4: a protein secondary structure prediction server, *Nucleic Acids Res.*, **43** (2015), W389–W394.

Supplements

We have published the code on github. The link is: https://github.com/xuying-6/Multi-scale-multi-task-network-model/

AIMS Press