**International Journal of Biology and Life Sciences**
ISSN: 2957-9511 | Vol. 6, No. 3, 2024

# Protein Secondary Structure Prediction based on Multi-scale Convolution and Transformer

**Shiwei Yang, Xiaozhou Chen ***

School of Yunnan Minzu University, Mathematics and Computer Science, Kunming, Yunnan, China
* **Corresponding author**: Xiaozhou Chen (Email: chxiaozhou@163.com)

**Abstract:** As an important field of natural science, bioinformatics reveals various complex biological facts. The task of protein structure prediction has always been an important research topic in bioinformatics, starting from the primary structure, namely the amino acid sequence, to predict the secondary structure of proteins. In this experiment, amino acid sequences were encoded in various ways, and the encoded data were input into the constructed deep learning network to predict the secondary structure of proteins. In this paper, an MCN-T network is constructed by combining CNN and Transformer. The powerful global feature capture and parallel processing capabilities of the Transformer are combined with the efficient local feature extraction and dimension reduction capabilities of CNN. This combination enables the model to perform well in protein secondary structure prediction tasks. In the experiment, the optimal form of the model is obtained after various parameter optimization experiments. The accuracy of the MCN-T network on the CB6133_filtered dataset and the CB513 dataset is 71.93% and 67.83%, respectively.

**Keywords:** Multi-scale Convolution; Structure Prediction; Transformer; Neural Network.

## 1. Introduction

As an important field of natural science, bioinformatics has been revealing various complex biological facts. The analysis of protein structure and function is an important research topic in bioinformatics. Proteins are macromolecule polypeptides, and macromolecules are mainly composed of various secondary structures of proteins, which are, in turn, composed of 20 basic amino acids through different arrangement combinations and chemical bonds. The secondary structure of proteins can be divided into eight irregular shapes: namely A (random coil), B (single $\beta$ fold), E (extended $\beta$ fold), G ($3_{10}$ helix), I ($\prod$-helix), H ($\alpha$ helix), S (high curvature ring), and T (turn), etc.

Li et al. proposed an end-to-end deep network (EEDN), which uses a CCNN (cascade convolutional neural network) and an RNN (recurrent neural network) to feed embedded sequence features and original contour features into multi-scale CNN layers with different kernel sizes to extract multi-scale locally relevant features. The secondary structure is predicted by combining local and global context features[1]. Zhang et al. proposed a new deep learning architecture combining a CNN, ResNet, and BiRNN. This model uses a recursive loop network to verify the protein structure categories of low- and high-dimensional datasets and applies the Stockwell transformation in the network [2]. Guo et al. used a hybrid deep learning framework, two-dimensional convolutional bidirectional recurrent neural networks (2C-BRNNs), which includes four models, including two-dimensional convolutional bidirectional GRUs. This model is very helpful for feature extraction [3]. Hu et al. used a set algorithm based on a bidirectional LSTM. They introduced an integration model. The technology consists of five sub-models, each of which contains two bidirectional LSTM layers to form an integrated model, and finally connects the bidirectional LSTM layer to the sub-model. The integrated model and sub-models are trained in parallel, and the performance of each model is observed [4]. Zhao et al. proposed a novel strategy for protein secondary structure prediction based on generative adversarial and convolutional neural networks. The network constructs an adversarial network to extract protein features and then combines the extracted features with the original Position-Specific Scoring Matrix (PSSM) data as the input of the convolutional neural network[5]. Zhao et al. proposed a convolutional neural network based on optimization, which uses sliding window technology and combines a convolutional neural network with a Bayesian optimization model to optimize the network framework and hyperparameters of the convolutional neural network[6]. All the above experiments show that convolutional neural networks have good performance in the field of protein secondary structure prediction. Therefore, this paper uses a multi-scale convolutional combined transformer to predict protein secondary structure.

## 2. Data Processing

### 2.1. Data Coding

The coding methods used in this experiment are PSSM spectrum coding, one-hot coding, physicochemical properties, and relative probability coding of eight protein secondary structures.

#### 2.1.1. PSSM Spectrum Coding

PSSM can be obtained by comparing protein sequences in a protein database. It was first proposed in 2008 by Steven Altschul et al.[7], and the PSSM spectrum contains more information about protein sequence evolution. Protein PSSM profiles can be obtained through an online tool called PSI-BLAST. When PSI-BLAST is used, the threshold is set to 0.001, and the number of iterations is set to 3. The dimension of the PSSM matrix for each protein obtained is 20*N, with 20 representing the twenty amino acid classes. Where N is the number of protein amino acids, the PSSM matrix based on protein sequence can be represented by the following formula:

$$PSSM = \begin{bmatrix} P_{1\to1} & P_{1\to J} & P_{1\to20} \\ P_{2\to1} & P_{2\to J} & P_{2\to20} \\ P_{3\to1} & P_{3\to J} & P_{3\to20} \\ \vdots & P_{I\to J} & \vdots \\ P_{M\to1} & P_{M\to J} & P_{M\to20} \end{bmatrix} \quad (1)$$

Therefore, each residue of the protein sequence in the obtained PSSM matrix has a total of 20 features, where each feature $P_{I\to J}$ represents the possibility of the residue mutating into the corresponding amino acid type.

#### 2.1.2. Onehot Spectrum Coding

There are 20 types of amino acids in multi-scale convolutional block proteins, abbreviated as A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, and Y. We encode them using a 21-dimensional orthogonal vector that contains only 0 and 1. Since some amino acids may not be recognized under certain circumstances, X is set to represent unknown amino acids that are not recognized, as shown in the following Table1:

**Table 1.** Amino acids are encoded in 21 dimensions

| | |
|---|---|
| **A :**100000000000000000000 | **M :** 000000000100000000000 |
| **C :** 010000000000000000000 | **N :** 000000000010000000000 |
| **D :** 001000000000000000000 | **P :** 000000000001000000000 |
| **E :** 000100000000000000000 | **Q :** 000000000000100000000 |
| **F :** 000010000000000000000 | **R :** 000000000000010000000 |
| **G :** 000001000000000000000 | **S :** 000000000000001000000 |
| **H :** 000000100000000000000 | **T :** 000000000000000100000 |
| **I :** 000000010000000000000 | **V :** 000000000000000010000 |
| **K :** 000000001000000000000 | **W :** 000000000000000001000 |
| **L :** 000000000100000000000 | **Y :** 000000000000000000100 |
| | **X :** 000000000000000000010 |

#### 2.1.3. Physical and Chemical Properties Coding

The formation of the secondary structure of proteins has an important relationship with the physical and chemical properties of amino acids [8][9], especially hydrophilic/hydrophobic properties. Physicochemical properties are divided into side chain type, polarity, charge, hydrophilicity, molecular weight, and frequency of occurrence in protein structures. It should be noted that nominal features such as side chain type and polarity must first be converted into numerical types and then normalized together with numerical features such as hydrophilicity. After adding features of physical and chemical properties, the constructed deep learning network can find associations between these properties and the protein secondary structure, thus providing more information for prediction. In addition, similar amino acid residue types have certain relative preferences for the formation of protein secondary structures. After considering the relative probability of occurrence of various amino acid residues in the secondary structure, the coding was carried out [10].

### 2.2. Sliding Window Technique

Sliding Window technology [11] is a common algorithm widely used in image processing, computer vision, natural language processing, and other fields. It performs analysis and processing by moving a fixed-size window over the input data. The basic principle is to divide the input data into multiple small pieces and move these small pieces for analysis. Specifically, sliding window technology divides the input data into successive windows, each representing a fixed-size segment of data. Then, it gradually moves the window from the beginning of the input data until the entire data is covered. During the moving process, various operations such as feature extraction can be carried out on the data in each window. Since the primary structure of a protein is the amino acid sequence, this technique is suitable for predicting the secondary structure of a protein. The interaction between amino acid sequences can be well represented by the sliding window technique. Generally, in the protein secondary structure prediction task, the selected window size is odd. In the sliding window, the amino acid in the middle of the selected amino acid fragment is the secondary structure of the target amino acid to be predicted, and the adjacent amino acids are input into the network as influencing factors of the secondary structure.

### 2.3. Data Set

The datasets used in this paper are the public datasets CB6133 and CB513 [12]. The CB513 dataset is composed of 514 first-order protein sequences, while the CB6133 dataset consists of 6218 first-order protein sequences. To reduce data homology, the CB6133 dataset was screened (sequences with more than 25% homology with CB513 were excluded), and the screened dataset was named CB6133_filtered. This screened dataset contains 5534 protein sequences. In this paper, the CB6133_filtered dataset is divided into 80% training data, 10% validation data, and 10% test data. The CB513 dataset is also used as test data.

## 3. Network Construction

### 3.1. Multi-scale Convolution Block

Convolutional neural networks (CNNs) have been widely used in the processing of sequence data. The classic CNN is usually composed of multiple layers, including but not limited to the convolutional layer, pooling layer, and fully connected layer. CNNs mainly carry out convolutional operations through convolutional layers, that is, they use convolutional kernels (filters) to slide on input data to extract input features locally. Traditional CNNs often use convolutional kernels of the same size, and such parameter settings may limit the ability to capture different features. In this paper, a multi-scale convolutional block (ms_block) is constructed. The purpose is to capture various features of input data through convolution kernels of different scales. By setting convolution kernels of different sizes, features of different sizes and complexities can be well extracted, thus improving the performance of the model. The core part of the multi-scale convolution block designed in this experiment is to extract multi-scale features of input data through four parallel convolution paths. Each path uses a different convolution kernel size to capture information at a different scale, and each branch first uses a one-dimensional convolution to check for initial feature extraction of the input data. Next, further convolution operations are performed on the initially extracted features using convolution kernels of different sizes (s1, s2, s3, s4, s5) to capture features of different scales. These parallel convolution branches are concatenated along channel dimensions to form a high-dimensional representation with rich multi-scale features.

### 3.2. Transformer Block

The Transformer, as a deep learning model based on the attention mechanism, was first proposed in 2017 by Vaswani

et al. [13]. It is mainly used for natural language processing and performs well in sequential data processing. The Transformer abandons the circular structure of recurrent neural networks (RNNs) and relies entirely on self-attention mechanisms and feedforward neural networks to capture global information in the sequence. Its core components include a multi-head self-attention mechanism, a multi-layer feedforward network, location coding, and a residual connection. The introduction of the Transformer into the network can make use of its powerful parallel processing capability and the ability to capture long-term dependencies to compensate for the shortcomings of CNNs in long-term information, so that the model can better understand the complex dependencies in sentences. The Transformer block constructed in this study consists of a multi-head self-attention mechanism, a feedforward neural network, and a residual connection. The multi-head self-attention mechanism captures global dependencies in input sequences, while feedforward neural networks further handle feature representations. Each Transformer encoder block contains two layer normalization and residual connection steps to ensure a stable training process. By stacking multiple Transformer encoder blocks, the model can extract higher-level feature representations layer by layer.

# 4. Experimental Design

In this paper, an MCN-T network is constructed by combining a convolutional neural network with a transformer network. The network uses the Transformer's powerful global feature capture capability and parallel processing capability to make up for the shortcomings of the CNN in processing long-term dependency and global information. Meanwhile, the CNN can enhance the computing efficiency and feature representation capability of the Transformer through efficient local feature extraction and dimension reduction capabilities. This combination enables the model to perform well in protein secondary structure tasks.

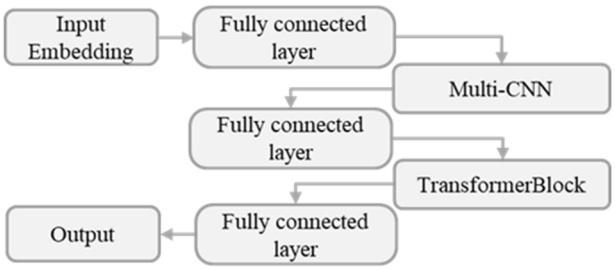


**Figure 1.** The MCN-T network

In the field of protein secondary structure prediction, many researchers choose to convert the data into one-dimensional vectors. This study chooses to convert the data into three-dimensional vectors as the input data for the model. After special data processing, each feature coding vector is obtained by a sliding window. Amino acids that need to be predicted are placed in the center of the sliding window. When the sliding window is located at the beginning and end of the sequence, zero filling is used to fill the vacancy in the window, and the corresponding amino acids are filled in other positions of the sliding window, as expressed in formula 2:

$$\tilde{S}_i = S_{i-\frac{x-1}{2}}, S_{i-\frac{x-3}{2}}, \cdots, S_{i+\frac{x-3}{2}}, S_{i+\frac{x-3}{2}} \quad (2)$$

For a protein sequence, the total dimension of the eigenvector is N*S*57, where N is the length of the amino acid sequence, S is the size of the sliding window, and 57 is the total dimension of PSSM spectral coding, one-hot coding, physicochemical properties, and logarithmic relative probability coding. In the data processing stage, all the feature codes are fused to make the data input into the multi-scale convolution block consistent.

In the multi-scale convolution module, five multi-scale convolution blocks are stacked in this study, and each convolution block is composed of the same convolution architecture, which is divided into four different branches. The LeakyReLU activation function is applied to all of them: the first branch is ir1, the second is ir2, the third is ir3, and the fourth is ir4. Three convolution layers are set up in ir1. The first layer uses convolution kernels of size 1*1, and the number of convolution kernels is 32. The second layer uses convolution kernels of size s1*s1, and the number of convolution kernels is 48. The third layer uses convolution kernels of size s1*s1, and the number of convolution kernels is 64.

The expression for the convolution operation is as follows:

$$ir_1 = F_{f\bullet s}(\mathrm{k}) \otimes X \quad (3)$$

$$ir_2 = F_{f\bullet s}(\mathrm{k}) \otimes ir_1 \quad (4)$$

$$ir_3 = F_{f\bullet s}(\mathrm{k}) \otimes ir_2 \quad (5)$$

The size of the convolution kernels set in the first layer of the other three branches is 1*1, and the number is [32, 48, 48]. The second layer uses convolution kernels of size [s2*s2, s3*s3, s4*s4], and the number of convolution kernels is [48, 64, 64], respectively. The third layer uses convolution kernels of size [s3*s3, s4*s4, s5*s5], respectively, and the number of convolution kernels is [64, 96, 96]. After the convolution architecture, the four branches are connected for multipath output, and the last dimension is connected to form a huge tensor, thus effectively combining the convolutional output of different scale kernel features to represent the input data in a richer way.

$$ir_{merge} = concat(ir1, ir2, ir3, ir4) \quad (6)$$

After the data is convolved with a 1*1 kernel, the output channel dimension is 256, and the linear activation function is used. Then, the Dropout operation is applied with a drop rate of 0.5. The purpose of this convolution and normalization process is to adjust the number of channels and integrate information from different paths.

$$ir_{conv} = Dropout(0.5)(BN(Conv1D()(ir_{merge}))) \quad (7)$$

Next, a shortcut path is designed, and the glorot_uniform initialization is applied to ensure uniform weight distribution. Finally, a residual connection is applied to the data to add the output of the main path and the output of the shortcut path, allowing the input information to be directly transferred to the output to alleviate the vanishing gradient problem.

$$X_{shortout} = BN(Con1D) \otimes X \quad (8)$$

$$out = \mathrm{LeakyReLU}(BN(add(ir_{merge}, X_{shortcut}))) \quad (9)$$

After the multi-scale convolution for fast data processing, an input transformer block with a multi-head attention mechanism is used.

$$Q = X, K = X, V = X \quad (10)$$

After the multi-scale convolution for fast data processing, an input transformer block with a multi-head attention mechanism is used.

$$A = MHAtt(Q, K, V) \tag{11}$$

$$A_{drop} = Dropout(A, r) \tag{12}$$

$$Z_1 = LN(X + A_{drop}) \tag{13}$$

$$Y = LN(Z_1 + Dropout(Dense(relu(Dense(Z_1, ff\_dim)), embed\_dim), r)) \tag{14}$$

In the transformer block, each step includes batch normalization and Dropout operations to ensure model stability.

# 5. Experimental Optimization and Analysis

## 5.1. Evaluation Index

In this experiment, the Q score[14] was used to evaluate the performance of the prediction model. This evaluation method is based on the proportion of the predicted amino acid residues to the predicted amino acids. The specific mathematical expression is as follows:

$$Q_c = \frac{1}{res}\sum_{i=1}^{c} T_{ii} \tag{15}$$

In the expression, $c$ indicates that the number of labels can be 3 or 8, corresponding to three-state structure prediction and eight-state structure prediction, respectively. $T_{ii}$ indicates the number of amino acid residues correctly predicted in the $i$ state, and $res$ represents the total number of amino acid residues.

Recall rate, accuracy, F1 score were also used to evaluate the experiment:

The formula for Recall is as follows:

$$Recall = \frac{TP}{TP + FN} \tag{16}$$

The precision formula is as follows:

$$Precision = \frac{TP}{TP + FP} \tag{17}$$

Where $TP$ is the number of positive samples correctly predicted by the model, $FP$ is the number of positive samples incorrectly predicted by the model as negative samples, and $FP$ is the number of negative samples incorrectly predicted as positive samples.

## 5.2. Optimization and Analysis

In this study, various comparative experiments were carried out with different parameters, network structures, and learning rates to obtain the best accuracy for the network. The sliding window technology was used to divide the data, and different experimental structures were obtained by using different window sizes. In order to test the influence of different window sizes on the experimental results, five different window sizes of 11, 13, 15, 17, and 19 were used, while other experimental parameters were kept unchanged.

As can be seen from Table 2, when the window size is 17, the experimental effect is the best, and the accuracy of the CB6133_filtered dataset and the CB513 dataset is 72.25% and 67.58%, respectively. When the window size is 11, the experimental effect is the worst. Only 70.04% and 65.99% accuracy were achieved on the two datasets.

**Table 2.** Experimental results of sliding Windows with different sizes

| Window size | CB6133_filitered dataset % | CB513 dataset % |
|---|---|---|
| **11** | 70.04 | 65.99 |
| **13** | 71.14 | 67.28 |
| **15** | 71.59 | 67.47 |
| **17** | 72.25 | 67.58 |
| **19** | 71.48 | 67.24 |

In MCN-T networks, the number of multi-scale convolution blocks determines the complexity of the model and also determines the number of parameters in the model. Therefore, the number of multi-scale convolution blocks has a great impact on the final result. Thus, this experiment included debugging of different numbers of convolution blocks, and the debugging results are shown in Table 3.

**Table 3.** Experimental results under different convolution blocks

| Number of convolution blocks | CB6133_filitered dataset % | CB513 dataset % |
|---|---|---|
| **3** | 71.30 | 66.96 |
| **4** | 71.04 | 67.26 |
| **5** | 71.93 | 67.83 |
| **6** | 70.60 | 67.33 |
| **7** | 71.70 | 67.52 |
| **8** | 71.48 | 67.32 |

It can be seen from Table 3 that when the number of convolutional blocks is 5, the experimental effect is the best, and the accuracy of the CB6133_filtered dataset and the CB513 dataset is 71.93% and 67.83%, respectively. When the number of convolutional blocks is 3, the experimental effect is the worst. Only 71.30% and 66.96% accuracy were achieved on the two datasets.

It can be seen that when there is a small number of convolutional blocks, as the number of added convolutional blocks increases, the model parameters increase and the depth of the model increases, enabling the network to better extract the features of the data. However, too many convolutional blocks make the model too complex and redundant, thus reducing the model's generalization ability and performance.

**Table 4.** Experimental results under different Transformer blocks

| Number of Transformer blocks | CB6133_filitered dataset % | CB513 dataset % |
|---|---|---|
| **2** | 71.93 | 67.83 |
| **3** | 71.79 | 67.61 |
| **4** | 70.25 | 65.90 |
| **5** | 71.32 | 67.32 |

Similar to the number of convolutional blocks, the number of Transformer blocks also has an impact on the depth and performance of the model. Therefore, this experiment conducted a comparative test on the number of Transformer blocks and the number of attention heads. The results are shown in Tables 4 and 5.

**Table 5.** Experimental results under different multi attention heads

| Focus more on the number of heads | CB6133_filitered % | CB513 % |
|---|---|---|
| **3** | 71.23 | 66.19 |
| **4** | 71.93 | 67.83 |
| **5** | 71.73 | 67.28 |
| **6** | 71.44 | 67.30 |
| **7** | 71.50 | 67.46 |
| **8** | 71.53 | 67.32 |

After the test, it can be seen from Table 4 and Table 5 that the experiment achieves the best effect when the number of Transformer blocks is 2 and the number of multi-attention heads is 4. Accuracy of 71.93% and 67.83% was achieved on both datasets. When the number of Transformer blocks is 4 and the number of multi-attention heads is 3, the accuracy decreases to 70.25% and 65.90% on the CB6133_filtered dataset and 71.23% and 66.19% on the CB513 dataset. It can be seen from the experimental results that too many stacked Transformer blocks and excessive addition of multi-attention heads lead to a deterioration in model performance.

### 5.3. Learning Rate and Dropout

The selection of the learning rate can also affect the results of the experiment. In this experiment, the RMSProp optimizer [15] was used. The RMSProp optimizer was first proposed by Geoffrey Hinton, who introduced this optimization algorithm in a Coursera machine learning course lecture in 2012. RMSProp is widely used in the optimization of various deep learning models because of its excellent performance in handling deep learning tasks, especially its good convergence speed and stability in the training process. Various learning rates were tested in the experiment, as shown in Table 6:

**Table 6.** Experimental results under different learning rates

| learning rate | CB6133_filitered dataset | | | CB513 dataset | | |
|---|---|---|---|---|---|---|
| | Accuracy% | Precision | Recall | Accuracy% | Precision | Recall |
| **0.00001** | 71.01 | 0.7982 | 0.6296 | 67.00 | 0.7572 | 0.5792 |
| **0.0001** | 71.93 | 0.7792 | 0.6656 | 67.83 | 0.7417 | 0.6162 |
| **0.0005** | 71.01 | 0.7762 | 0.6057 | 67.03 | 0.7367 | 0.6057 |
| **0.001** | 58.65 | 0.7862 | 0.3883 | 55.50 | 0.7513 | 0.3445 |

In this experiment, we tested a variety of learning rates. When the initial learning rate was 0.00001, the accuracy of our model on the CB6133_filtered dataset and the CB513 dataset was 71.01% and 67%, respectively. With an initial learning rate of 0.0001, our model achieved an accuracy of 71.93% and 67.83% on both datasets. As the accuracy increased, the learning rate was increased again. When the initial learning rate was 0.0005, the accuracy of the two datasets was 71.01% and 67.03%, and the accuracy decreased. When the initial learning rate was 0.001, the accuracy of the two datasets dropped to 58.65% and 55.50%. Thus, we ultimately adopted an initial learning rate of 0.0001.

Due to the large dataset, the training time of the model is relatively long. In order to solve the overfitting problem, we added Dropout to the convolutional layer of the model, which discards some neurons after every training cycle. In our model, the selected dropout rate is 0.5.

## 6. Conclusion

In this experiment, in order to predict the secondary structure of proteins, we designed a network model combining multi-scale convolution and Transformer. We analyzed the model from multiple perspectives and finally obtained the optimal form of the model after multiple parameter optimization experiments. The accuracy of our MCN-T network on the CB6133_filtered and CB513 datasets is 71.93% and 67.83%, respectively. In this experiment, we propose and prove the possibility of combining a multi-scale convolutional neural network with a Transformer network for sequence labeling, and found the optimal combination of precision through fine-tuning parameters. In future work, more features can be added, which may effectively increase the accuracy of the model.


## Acknowledgments

Thanks to Professor Chen Xiaozhou for his guidance on the direction of my research and the significance of the research results, and the National Natural Science Foundation of China (Approval number :31460297) for the funding.


## References


[1] Li, Zhen, and Yizhou Yu. "Protein secondary structure prediction using cascaded convolutional and recurrent neural networks." arxiv preprint arxiv:1604.07176 (2016).

[2] Zhang B, Li J, Lü Q. Prediction of 8-state protein secondary structures by a novel deep learning architecture[J]. BMC bioinformatics, 2018, 19: 1-13.

[3] Guo Y, Wang B, Li W, et al. Protein secondary structure prediction improved by recurrent neural networks integrated with two-dimensional convolutional neural networks[J]. Journal of bioinformatics and computational biology, 2018, 16(05): 1850021.

[4] Hu H, Li Z, Elofsson A, et al. A Bi-LSTM based ensemble algorithm for prediction of protein secondary structure[J]. Applied Sciences, 2019, 9(17): 3538.

[5] Zhao Y, Zhang H, Liu Y. Protein secondary structure prediction based on generative confrontation and convolutional neural network[J]. IEEE Access, 2020, 8: 199171-199178.

[6] Zhao Yawu, Liu Yihui. Protein secondary structure prediction based on optimized convolutional neural networks [J]. Computer Applications and Software, 2021, 38(7): 147-152,166.

[7] Altschul S F, Gertz E M, Agarwala R, et al. PSI-BLAST pseudocounts and the minimum description length principle[J]. Nucleic acids research, 2009, 37(3): 815-824.

[8] Creighton T E. Proteins: structures and molecular properties[M]. Macmillan, 1993.

[9] Hsu C W, Lin C J. A comparison of methods for multiclass support vector machines[J]. IEEE transactions on Neural Networks, 2002, 13(2): 415-425.

[10] Wu Yuming. A new coding method for protein secondary structure prediction [J]. Industrial Control Computer, 2015 (4): 109-110.

[11] Majid Vafaeipour, Omid Rahbari, Marc A. Rosen,Farivar Fazelpour,Pooyandeh Ansarirad. Application of sliding window technique for prediction of wind velocity time series[J]. International Journal of Energy and Environmental Engineering, 2014,5(2-3).

[12] Zhao Y, Liu Y. OCLSTM: Optimized convolutional and long short-term memory neural network model for protein secondary structure prediction[J]. PLOS ONE, 2021, 16.

[13] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. Advances in neural information processing systems[J]. Advances in neural information processing systems, 2017, 30(2017).

[14] Jiang Q, Jin X, Lee S J, et al. Protein Secondary Structure Prediction: A Survey of the state of the art [J]. Journal of molecular graphics & modelling, 2017, 76:379-402.

[15] Zhang Tian-Ze, Li Yuan-Xiang, Xiang Zheng-Long, et al. Particle swarm optimization algorithm based on RMSprop [J]. Computer Engineering and Design, 2021, 42(3): 7.