Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 10, No. 3, 2024

Prediction of Protein Secondary Structure Using a Hybrid Convolutional Blocks with GRU Units

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: In order to better understand the function of proteins, protein research in the field of bioinformatics has always been an important issue, and protein structure prediction is also one of the research topics. Starting from the one-dimensional structure (amino acid sequence), it is a good method to predict and classify the secondary structure and adopt deep neural networks to solve sequence problems. We used convolutional blocks combined with gated recurrent units (GRUs) to predict protein secondary structure. Different from previous convolutional neural networks architectures, in this study, we used a mixture of two convolutional blocks of different scales combined with GRUs for sequence labeling for protein secondary structure prediction. Additionally, considering that the coding format in previous studies was too simple, this experiment also added the physical and chemical properties and the logarithmic relative probability of 8 types of secondary structure amino acids as the feature input, which improved the accuracy after the addition of features. Experiments show that our model has good performance in Q8 accuracy.

Keywords: Convolutional Block; GRUs; Secondary Structure.

1. Introduction

Proteins are the material basis of essential life activities and the main bearers of life activities. They can regulate antibodies, cell structural elements, motor elements, promote chemical reactions, and regulate cell activity. As we all know, there is a complex relationship between protein function and its structure; it can be said that the structure a protein presents determines the function it achieve[1]. Therefore, in proteomics, how to make the prediction of protein secondary structure more efficient and highly precise has become a hot topic[2]. In the past, researchers usually used X-ray crystallography, nuclear magnetic resonance spectroscopy, and frozen electron microscopy experiments to predict the secondary structure of proteins. However, these experiments are rigorous and expensive, and can be interfered with by various factors, especially when the amino acid fragments of some proteins cannot pass certain parts of the experiment, usually resulting in less-than-ideal result[3].

However, with the continuous development and advancement of deep learning and artificial intelligence, computer science has intermingled, attracting a large number of researchers to use deep learning methods to study the relationship between protein sequences and their structures[4]. During the process of predicting protein structures, it is possible to predict the secondary structure sequence of each position in the protein's primary structure. In protein secondary structure prediction tasks, there are typically two types of predictions: eight-class prediction (Q8) and three-class prediction (Q3). Compared to Q3, Q8 can reveal more detailed information about protein structure. Therefore, this article focuses on the Q8 prediction of protein secondary structure[5].

In the study of protein secondary structure prediction, a large number of researchers use computers to build deep learning models. In the research process, the task of predicting positions from the primary structure sequence and generating new sequences is referred to as sequence labeling. With the continuous development of research, various machine learning model methods such as support vector machine [6], recurrent neural networks[7], and convolutional neural networks (CNNs)[8] have been widely used in various prediction experiments of protein structure. Among these, convolutional neural networks and recurrent neural networks are favored for their excellent performance on sequence data.

As a special kind of sequence data, protein amino acid sequences can be utilized to extract local context features by convolutional neural networks. Studies in recent years have shown that this approach has achieved good results in protein secondary structure prediction. However, convolutional neural networks mainly focus on local spatial structures while ignoring the temporal order dimension. Therefore, recurrent neural networks are introduced in this paper. By incorporating cyclic connections, sequential information can be captured and processed, enhancing the network's ability to handle sequence data. In this study, both convolutional blocks from convolutional neural networks and GRUs[9], a variant of recurrent neural networks, are employed to predict the secondary structure of proteins.

2. Secondary Protein Structure

The prediction of protein secondary structure usually begins with the primary form of the protein, known as the primary structure, which provides the amino acid composition for the protein. The secondary structure can also be briefly described as the relative spatial arrangement of amino acids, where 20 amino acids are connected together in a specific order to form a protein chain. These protein chains exhibit different spatial structures in various folded spatial forms. The following Table1lists the names, abbreviations, and symbols of the 20 amino acids[10]:

Table 1. Names, abbreviations and symbols of 20 amino acids.

Name	Abbreviation	Symbol	Name	Abbreviation	Symbol
Alanina	Ala	A	**Methionine**	Met	M
Cysteine	Cys	C	**Asparagine**	Asn	N
Aspartic Acid	Asp	D	**Proline**	Pro	P
GlutamicAcid	Glu	E	**Glutamine**	Gln	Q
Phenylalanine	Ph	F	**Arginine**	Arg	R
Glycine	Gl	G	**Serine**	Ser	S
Histidine	His	H	**Threonine**	Thr	T
Isoleucine	Ile	I	**Valine**	Val	V
Lysine	Lys	K	**Tryptophan**	Trp	W
Leucine	Leu	L	**Tyrosine**	Tyr	Y

In the classification task, the secondary structure of proteins is divided into eight categories[11]:

L (random coil), B (single β fold), E (extended β fold), G (3_{10} helix), I ($\prod$-helix), H (α helix), S (high curvature ring), and T (turn)and other structures, collectively referred to as the Q8 classification introduced above. The corresponding forms of the primary structure and secondary structure are as follow figure 1:

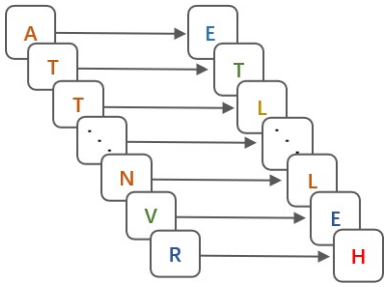

Figure 1. Protein primary structure corresponds to secondary structure.

3. Experimental Data and Coding

3.1. Experimental Data

In this paper, the CB6133 and CB513 datasets generated by the Pisces cullpdb are used as input data and training data for the model[12]. The CB513 dataset consists of 514 protein first-order sequences[13], and the CB6133 dataset consists of 6218 protein first-order sequences. Due to the large amount of sequence homology redundancy between and within the datasets, the accuracy of the prediction algorithm may be overestimated[14]. To reduce the homology within the data, the CB6133 dataset is screened (excluding sequences with homology greater than 25% to CB513). The filtered dataset, named CB6133filtered, contains 5534 protein sequences. In this paper, the CB6133filitered dataset is divided into 80% training data, 10% validation data, and 10% test data. The length of each protein sequence is 700, and 514 data from the CB513 dataset are also used as test data. The length of each protein sequence is 700, and the dataset contains 21-dimensional amino acid residues, 21-dimensional orthogonal codes as input features, and unique thermal codes of eight protein secondary structures as classification labels. In order to improve the accuracy of structure prediction, the physical and chemical properties of various amino acids and the logarithmic relative probabilities of eight secondary structure amino acids were added in this paper, and these were encoded to carry out feature fusion with the input features described above.

3.2. PSSMS and Orthogonal Coding

Position specific scoring Matrix (PSSMs) was first proposed by Steven Altshchulden et al[15]. In 2008. PSSMs were obtained from the comparison of multiple sequences, which can extract protein sequence features well, and are generated by PSI-BLAST[16] searching for protein sequences related to the evolution of the queried protein sequence. After several queries, sequences with more evolutionary information were found until the obtained sequence contains no more evolutionary information. The following is a PSSM matrix based on protein sequences:

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

In the given PSSM matrix, $P_{I\to J}$ represents the probability that the I th amino acid residue replaces the J th amino acid residue in the protein sequence. After the PSSM matrix is generated through PSI-BLAST iteration, it is necessary to use functions to normalize the generated matrix because the generated matrix is an integer matrix. In the normalization process, the sigmoid function is generally used. After processing, the value of the matrix will be remapped to the interval [0,1]. The sigmoid function calculation formula is given below:

$$F(x) = \frac{1}{1+e^{x}} \quad (2)$$

The orthogonal coding of proteins [17] is to represent 20 protein amino acid sequences and unknown amino acid sequences as 21-dimensional orthogonal vectors containing only 0 and 1, among these, 20 known sequences are represented as A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, and Y as seen above. Since the codes of any two amino acids are orthogonal, it can be seen that the inner product between each amino acid is 0. In addition to the 20 amino acids introduced above, X is added to represent the 21st unknown amino acid sequence because the simulation experiment sometimes cannot determine the specific type of a certain amino acid. The following figure2 shows the orthogonal codes corresponding to 20 amino acids and X:

A:10000000000000000000 M:00000000010000000000
C:01000000000000000000 N :00000000001000000000
D:00100000000000000000 P :00000000000100000000
E:00010000000000000000 Q :00000000000010000000
F:00001000000000000000 R :00000000000001000000
G:00000100000000000000 S :00000000000000100000
H:00000010000000000000 T :00000000000000010000
I :00000001000000000000 V :00000000000000001000
K:00000000100000000000 W:00000000000000000100
L:00000000010000000000 Y :00000000000000000010
X :00000000000000000001

Figure 2. Amino acids correspond to orthogonal coding.

4. Network Construction

4.1. SM Convolution Block

Convolutional blocks, as the basic constituent unit of convolutional neural networks, are usually composed of convolutional layers, activation function layers, and other optional layers (which vary from data to data). These blocks are designed to extract features from data and pass them to the next layer of the network for further feature extraction or processing. Convolutional blocks typically begin with the convolutional layer. Where the convolution kernel (i.e. filter) is used to convolve the input data and extract its feature.

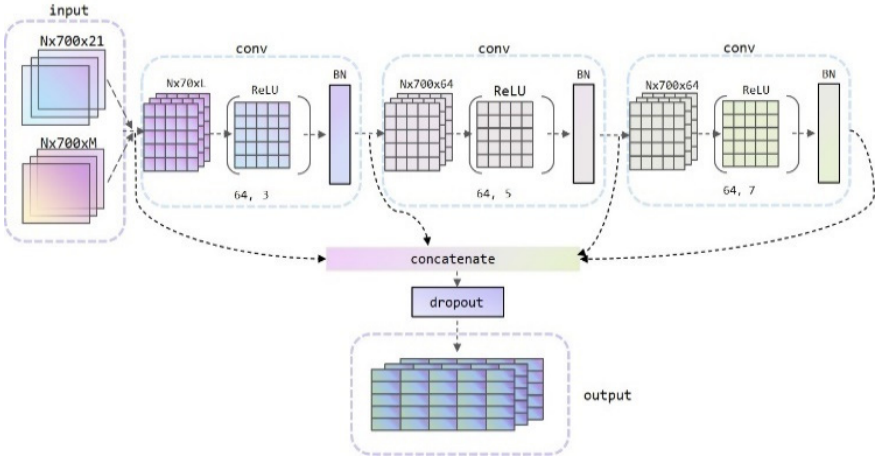

Figure 3. MSC Convolutional block

After the convolutional layer, in order to introduce nonlinearity, the activation function layer is usually added, and the commonly used activation functions include ReLU(Rectified Linear Unit), Tanh, Sigmoid, etc. In order to improve the stability and performance of the model, both the BN layer and the Dropout layer are added during the experiment. Batch normalization can retain more feature information even with a smaller data volume.

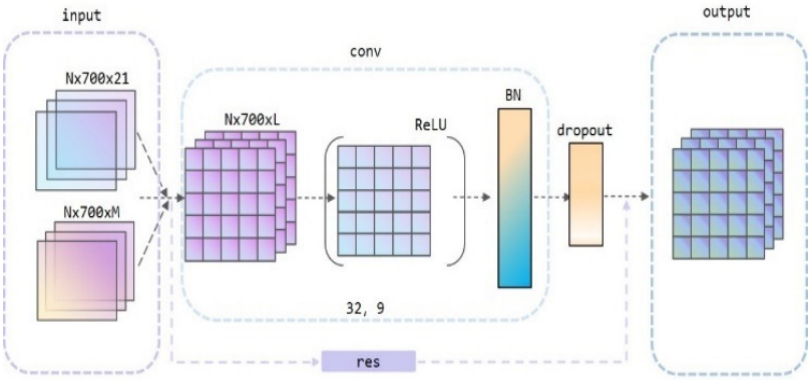

Figure 4. SSC Convolutional block.

In addition, it can also mitigate the problems of gradient disappearance and explosion, thus contributing to better and more stable training of the model. In this experiment, two types of convolution blocks are implemented in the network: namely SSC convolution block (as figure3) with single scale convolution block SS-CNN (as figure4) and MSC convolution block with multi-scale convolution block MS-CNN. By combining the two, this paper refers to it as the SM-CNN network, hereafter denoted as SMC network. The data processed by the SMC network exhibits certain robustness, which can improve the generalization ability of the model.

4.2. Bidirectional GRU

Recurrent neural networks (RNNs) have been widely used in sequence data processing. Experiments have utilized RNNs to address the time dependence of sequence. However, RNNs may struggle to deal with long-term dependence due to the problem of gradient explosion and disappearance. In the development of sequence data processing with RNNs, to tackle the issue of long-term dependence, the long-short memory networks (LSTM) were introduced.

As an enhanced version of RNNs, the LSTM network incorporates additional memory units and gates. These memory units and gates allow for selective forgetting or storage, enabling effective handling of long-term dependencies. However, since the design of LSTM network is more complex and the parameters used are very large, this paper introduces gated recurrent unit (GRU) network. As an improvement of LSTM network, the GRU network also features memory units and gates.

In comparison to the LSTM network, the GRU network integrates an update gate and a reset gate, reduces certain

LSTM parameters, optimizes its network structure, simplifying the network structure to a relatively simpler form, and exhibits better generalization ability with less training data, attributed to the improvement of its gate control mechanism.

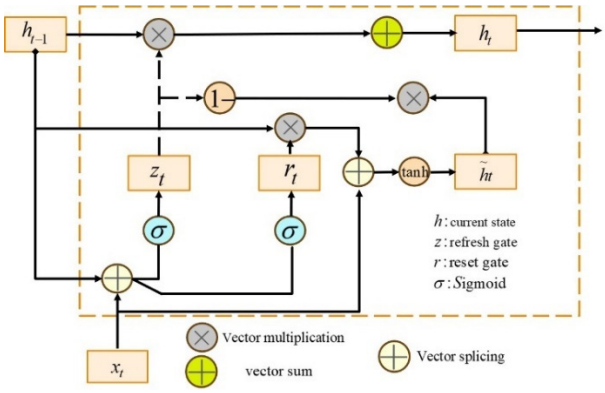

Figure 5. GRU cell.

GRU (as figure5) contains update gate (z_t) and reset gate (r_t). z_t determines the amount of saving the previous memory to the current memory unit, and r_t determines the ratio of new information to the previous memory unit. These two gates determine what information can be used as the output of GRU unit. The special feature of the gated unit is that it can store long-term sequence information, ensuring that the information will not be erased over time.

The update gate helps the model determine how much information can be passed on to the next step, or how much information in the previous memory unit needs to be passed on to the future. Where x_t is the input vector at time t, that is, the t component of the data series x, which is linearly transformed (that is, multiplied by the weight matrix), $h(t-1)$ contains the information of the previous memory unit, that is, the information at time $t-1$, which also goes through a linear transformation, and then combines the information of these two parts and sends it into the sigmoid function. To map the result between 0 and 1, here is the formula for this step:

$$z_t = \sigma\left(w^{(z)}x_t + U^{(z)}h_{t-1}\right) \tag{3}$$

The reset gate determines how much information needs to be forgotten in this memory unit. The same $h(t-1)$ and x_t need to undergo a linear transformation, and then send the information of both parts to the sigmoid function output:

$$r_t = \sigma\left(w^{(r)}x_t + U^{(r)}h_{t-1}\right) \tag{4}$$

After the reset gate is processed, the new memory information will store some relevant information in the past, and the Hadamard product (that is, the product of corresponding elements) is performed between r_t and $h(t-1)$, where the Hadamard product will determine the information that needs to be retained and forgotten at this moment, because the reset gate vector is composed of 0 to 1. So it will measure the value of the gated open, if the element gated value is 0, it means that the information carried by the element will be completely forgotten:

$$\tilde{h}_t = \tanh\left(wx_t + r_t \odot Uh_{t-1}\right) \tag{5}$$

In the final memory moment of this moment, the update gate determines the information $\tilde{h}(t)$ of the current memory moment and the information that needs to be collected in the previous $h(t-1)$ moment. z_t activated by the activation function will also control the input of information in the form of gate control. The Hadamard product of z_t and $h(t-1)$ represents the final memory information retained after screening. This information, together with the final memory information for the current memory moment (t moment), forms the final output of the GRU unit.

4.3. Integrate Deep Learning Models

The SMC-BiGRU network used in this work can effectively predict PSS. This network combines single-scale convolutional blocks, multi-scale convolutional blocks, residual networks, and bidirectional GRU neural networks to extract local information from input data by combining single-scale and multi-scale convolutional blocks. After the information extraction, data are input to the Bi-GRU layer to further extract long-range sequence data. This enables the extraction of protein sequence information more completely.

5. Experimental Design

Since the length of each sequence in the data set is different, each sequence is normalized to a length of 700. If the protein sequence is shorter than 700, fill it with zeros. If it exceeds 700, it is truncated to 700 characters. Truncated characters will continue to be filled to 700. To better preserve amino acid characteristic information, two data input channels were set up in the experiment. Channel one carried the orthogonal coding information of amino acid residues, and this orthogonal coding information was recorded as vector $w_1 = (0,\cdots,0,i,0,\cdots,0)$. An eigenmatrix P_1 with the shape of N*700*21 was obtained, composed of 20 amino acids and the assumed unknown amino acid X (N represents the sequence quantity). Channel 2 carries amino acid PSSM matrix information and other coding information. In this channel, the PSSM code of each amino acid is recorded as a 20-dimensional vector $w_2 = (P_i1,\cdots,P_i20)$. The meaning of P_i1 can be seen from the previous introduction, where P_i20 represents the probability that the i amino acid residue replaces the first amino acid residue in the protein sequence. This holds true for different types of input data as well. To maintain consistent data, feature types when inputting into the SMC layer, vector mapping is carried out on the data from input channel 1 at the feature fusion layer, transforming sparse data into dense data. The resulting dense data is then fused with the data from channel 2.

From the previous section, it can be observed that the SMC layer contains two convolutional blocks, namely the MSC convolution block and the SSC convolution block. The MSC convolution block consists of a convolutional layer, activation function, and batch standardization processing (BN layer). To capture information from different scales of the input sequence more effectively, three convolutional kernels of different sizes are set in the convolutional layer. Additionally,

to maintain the length of the input and output sequences, the ``same" mode is used for padding. Subsequently, the ReLU activation function is applied to nonlinearly correct the output of the previous convolutional layer. Batch standardization processing (i.e., BN layer) is utilized to stabilize the data, and the outputs of the three convolutional kernels of different sizes are merged with the original output to form a richer feature set. Finally, a dropout layer with a dropout rate of 0.5 is added after the feature fusion layer to prevent overfitting.

The SSC convolutional block also includes a convolutional layer, activation function, batch normalization processing (BN layer), and Dropout layer. The convolutional layer employs a one-dimensional convolutional layer to capture local information from the protein sequence, and the ``same" mode is used for padding. Following this, the ReLU activation function and batch normalization processing, similar to the MSC convolution kernel, are applied. A dropout layer with a dropout rate of 0.5 is added to prevent overfitting. In contrast to the MSC convolution kernel, a residual connection layer is included in order to mitigate the problem of gradient disappearance. This layer can retain initial data while processing new information, thereby enhancing the completeness of the sequence information obtained by the network. By employing two different connection methods, the sequence feature information can be extracted more comprehensively.

Once the data extracted by the SMC network is input into the Bi-GRU network, the Bi-GRU network extracts feature from the input data in both forward and backward directions, and subsequently concatenates the feature outputs obtained from both directions. Additionally, to reduce overfitting, an L2 regularization term is applied in the Bi-GRU layer. In this model, a regularization parameter with a value of 0.2 is chosen. Furthermore, temporal distribution dropout is applied in this layer, where a portion of neurons at each time step is independently dropped at a dropout rate of 0.5.

After the Bi-GRU layer processing, the data is converted into bidirectional output, and then inputted into the fully connected Dense layer. In this layer, the TimeDistributed function is used to apply the fully connected layer to the output of each time step, referred to as the TDDense layer, with ReLU being used as the activation function. The Dropout rate for time distribution is also applied, with a dropout rate of 0.5. In the final output layer, a label is defined to predict each time step, with the output dimension of this layer being the number of labels. The activation function is specified as SoftMax to convert the output data into a probability distribution of the class. The Glorot initialization weight is applied to evenly distribute the initialization weight, and the initializer of the offset term is specified as zero initialization.

6. Evaluation and Optimization

6.1. Evaluation Index

To evaluate the performance of an experimental model in forecasting, Q8 accuracy is often used as a measure. The higher the accuracy of Q8, the stronger the predictive ability of the model.

Q8 accuracy calculation formula:

$$Q8 = \frac{\sum_{i=1}^{8} P_{ii}}{N} \times 100 \tag{6}$$

where P_{ii} represents the correctly predicted number of amino acids, N represents the total number of amino acid residues.

6.2. Loss Function

The categorical_crossentropy loss function is selected for this experiment, as it is a type of crossentropy loss function that is suitable for multi-classification problems. It is chosen to adapt to the data type and to improve the accuracy of predictions. The cross-entropy loss function is often used to measure the difference between probability distributions. Since the Q8 classification problem entails multiple classification problems, the output format of the model consists of a probability distribution that represents the probability of each class. The categorical_crossentropy function calculates the crossentropy between the output probability distribution and the probability distribution of the actual labels, as follows:

$$\text{Loss} = -\sum_{i} y_{\text{true},i} \bullet \log\left(y_{\text{pred},i}\right) \tag{7}$$

Where $y_{ture,i}$ is the probability of the i-class of the true label, and $y_{pred,i}$ is the probability of the i-class predicted by the model.

6.3. Optimization Algorithm

Optimization algorithms are generally used to adjust the parameters of neural networks to minimize the loss function. Commonly used optimization algorithms include SGD, AdaGrad, Adam, and RMSprop, etc. In this experiment, RMSprop is selected as the optimization algorithm for neural networks. As a variant of the SGD algorithm, RMSprop can solve the problem of learning rate in the standard gradient descent method. The RMSprop algorithm can adjust the learning rate adaptively by maintaining the moving average of the gradient square. Its update rules are as follows:

Calculate the exponential weighted average motion of gradient square method:

$$v_t = \beta \bullet v_{t-1} + (1-\beta) \bullet \left(g_t \odot g_t\right) \tag{8}$$

Represents the element performance operation, v_t is the moving average of the gradient squared at time t, β is the decay rate, g_t is the current gradient.

Use the square root of the normal gradient as the scaling factor of the learning rate:

$$\Delta w_t = \frac{\eta}{\sqrt{v_t + \varepsilon}} \odot g_t \tag{9}$$

η is the learning rate and ϵ is a constant to avoid zero errors.

Update parameters:

$$w_{t+1} = w_t + \Delta w_t \tag{10}$$

Through the above updating steps, the RMSprop algorithm can gradually optimize the parameters of the model and finally minimize the loss function to enhance the model's performance.

6.4. Regularization

L2 regularization is to add a penalty term to the loss function, which is in the form of the original loss function and the sum of squares of the weight multiplied by a

regularization parameter, the specific formula is as follows:

$$\text{Loss} = \text{OriginalLoss} + \lambda \times w_i^2 \quad (11)$$

Where OriginalL oss is the original loss function, that is, the model objective function, w_i is the weight parameter of the model, and λ is the regularization parameter, which is used to control the intensity of regularization.

7. Results and Discussion

In this study, the model is adjusted by finding the best combination of parameters. For this purpose, this paper explores multiple parameters, including the size and number of convolutional nuclei and the number of hidden layers in the convolutional layer, and investigates the influence of learning rate on accuracy. In this experiment, the combination of SSC layer and MSC layer is called an SMC block. The number of SMC blocks determines the number of convolution layers of the model, and the convolution kernel along with the number of convolution layers jointly determines the size of the final receptive field. Therefore, determining the number of SMS blocks in the model and the size of the convolution kernel have a significant influence on the final prediction result. In order to explore the effect of the convolution layer on the accuracy of the model, the model with the number of SMC blocks set at 1, 2, 3, 4, and 5 was tested while keeping the model parameters unchanged. As shown in Table 2, the model achieves the best performance when the number of SMC blocks is 3 layers, with an accuracy rate of 72.4% on the CB6133_filtered test set and68.5% on the CB513 test set.

Table 2. Effects of convolutional layers on model performance.

SMCblocks	CB6133_filtered accuracy (%)	CB513 accuracy (%)
W1	70.5	67.2
W2	72.0	68.2
W3	72.4	68.5
W4	70.3	67.6
W5	71.9	67.7

After exploring the influence of SMC blocks on model accuracy, the influence of convolution kernel size on model performance was explored with the number of SMC blocks guaranteed to be 3, and the model accuracy under each convolution was analyzed while other parameters remained unchanged. As can be seen from Table 3, when the convolution kernel of SSC layer and MSC layer is set to 1 layer, the model shows poor performance, and the accuracy rate is 64.3% on the CB6133_filtered test set and 59.8% on the CB513 test set. When the convolutional nuclei of the SSC layer are set to 9 and the three convolutional nuclei of the MSC layer are set to 3, 5, and 7 respectively, the model performs the best, with an accuracy rate of 72.5% on the CB6133_filtered test set and 68.8% on the CB513 test set. After that, the accuracy of Q8 was reduced by increasing the size of convolutional nuclei. Through experimental comparison, it was found that the best performance was achieved when the convolutional nuclei of SSC layer were 9 and the three convolutional nuclei of MSC layer were set to 3, 5, and 7 respectively.

After determining the structure and parameters of the model, we tested different initial learning rates and set up a learning rate attenuation mechanism for model optimization in this experiment.

Table 3. Influence of convolution check on model performance.

SSC blocks	SMC blocks	CB6133_ filtered accuracy(%)	CB513 accuracy(%)
1	**1,1,1**	64.3	59.8
3	**3,3,3**	71.7	67.4
5	**5,5,5**	72.1	68.0
7	**7,7,7**	71.5	68.2
9	**9,9,9**	71.6	68.1
11	**11,11,11**	72.3	68.3
13	**13,13,13**	72.1	68.4
15	**15,15,15**	72.0	67.6
1	**3,5,7**	70.9	67.5
3	**5,7,9**	70.6	67.1
5	**7,9,11**	72.4	68.2
7	**9.11.13**	71.7	68.0
7	**1,3,5**	70.6	67.6
9	**3,5,7**	72.8	68.5
11	**5,7,9**	72.6	68.2
13	**7,9,11**	71.6	68.0
15	**9,11,13**	70.0	66.6

The optimizer we used was RMSprop, and a function called ``scheduler" was designed to adjust the learning rate. It can decide whether to adjust the learning rate according to the current number of training cycles and return the new learning rate. We also defined a learning rate scheduler, which is used to call the function ``scheduler". In this experiment, we adopted the same learning strategy for all initial learning rates. At the end of the 60th training cycle, the learning rate of the model is halved. It can be seen from the Table 4 that only 62.2% and 58.9% Q8 accuracies are obtained on the CB6133_filtered dataset and CB513 dataset when the initial learning rate is 0.00001. With the initial learning rate set to 0.0001, the Q8 accuracy rates are 68% and 65.7% for the CB6133_filtered dataset and CB513 dataset, respectively. Through comparison, it was found that increasing the learning rate significantly helped improve the accuracy rate. After adjusting the learning rate, accuracy rates of 72.5% and 68.8% were achieved on the CB6133_filtered dataset and CB513 dataset, respectively, when the learning rate was set to 0.0005. After increasing the learning rate to 0.005, it was found that the accuracy of Q8 began to decline. Through experimental comparison, it was determined that the learning rate of 0.0005 yielded the best results.

Table 4. Influence of learning rate on model performance.

Learning rate	CB6133_filtered accuracy (%)	CB513 accuracy (%)
0.00001	62.2	58.9
0.0001	72.2	68.3
0.0005	72.5	68.8
0.001	71.0	68.2
0.005	66.6	63.3

After adjusting the parameters, the experiment utilized multi-feature fusion to process the data. We tested the model's accuracy under different features, primarily PSSM spectral coding and orthogonal coding, and fused these two features. As shown in Table5, PSSM spectral coding contributed more significantly and performed more prominently compared to orthogonal coding. The experiment also demonstrates that the performance of all fusion features surpasses that of any single

feature, with the Q8 accuracy of all features post-fusion being the highest. This indicates that feature fusion positively aids in improving accuracy.

Table 5. Q8 accuracy rate under various feature coding forms.

Feature coding form	CB6133_filtered accuracy(%)	CB613 accuracy(%)
PSSM spectral coding combines	71.6	67.4
quadrature encoding	59.6	56.1
PSSM spectral coding combined with orthogonal coding	71.7	67.7
PSSM spectral coding combines physicochemical properties with logarithmic relative probability	71.9	68.1
Orthogonal coding combines physicochemical properties with logarithmic relative probability	59.7	56.5
All characteristics	72.5	68.8

Finally, the accuracy of the CB513 dataset is compared with other similarity models proposed by researchers(as Table 6), as follows: DeepSeqVec[18], DeepProf+SeqVec[18], Deep-CNF [19], Multi-scale CNN One-Hot Encoded[8], MUST-CNN[20], Bi-RNN Single Model[21], and Fine-tuned CNN[22] with Fine-tuned CNN-SVM[22].

Table 6. Accuracy of similarity model on CB513 data set.

Model	Q8 (%)
DeepSeqVec [18]	62.5 ± 0.6
DeepProf+SeqVec	66.0 ± 0.5
Deep-CNF	68.3
Multi-scale CNN One-Hot Encoded	68.3
MUST-CNN	68.4
Bi-RNN Single Model	68.5
Fine-tuned CNN	68.711
Fine-tuned CNN-SVM	68.735
SMC-BiGRU	68.86

8. Conclusion

After experimental analysis and comparison, the SMC-BiGRU model we employed proves capable of predicting the secondary structure of proteins with commendable performance. Additionally, the multi-feature fusion method we adopted effectively enhances the accuracy of model prediction. We fine-tuned various parameters of the convolutional block and GRU network to identify the combination yielding the highest precision. The convolutional block was designed to capture dependencies within each residue in the amino acid sequence effectively, while the bidirectional GRU network further captures long-term dependencies between sequences. Our model achieved a Q8 accuracy of 72.5% on the CB6133_filtered accuracy dataset and 68.86% on the CB513 dataset. This experiment confirms the effectiveness of our 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] M. Zamani and S. C. Kremer, "Protein secondary structure prediction through a novel framework of secondary structure transition sites and new encoding schemes," 2016 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), Chiang Mai, Thailand, 2016, pp. 1-7.

[2] Yuedong Yang, Jianzhao Gao, Jihua Wang, Rhys Heffernan, Jack Hanson, Kuldip Paliwal, Yaoqi Zhou, Sixty-five years of the long march in protein secondary structure prediction: the final stretch?, Briefings in Bioinformatics, Volume 19, Issue 3, May 2018, Pages 482–494.

[3] Y, Y., J, G. & J., W. Sixty-five years of the long march in protein secondary structure prediction: the final stretch. Briefings Bioinforma. 19, 482–494.

[4] Martin E. M. Noble et al.Protein Kinase Inhibitors: Insights into Drug Design from Structure.Science303,1800-1805 (2004).

[5] Srinivasa, K. G., Siddesh, G. M. & Manisekhar, S. Statistical modelling and machine learning principles for bioinformatics techniques, tools, and applications (Springer Nature, 2020).

[6] Kabsch, W. & Sander, C. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolym. Orig. Res. on Biomol. 22, 2577–2637, (1983).

[7] Shutong Yang, Yuhong Wang, Kennie Cruz-Gutierrez et al. Localnet: A Simple Recurrent Neural Network Model for Protein Secondary Structure Prediction Using Local Amino Acid Sequences Only, 07 January 2021, PREPRINT (Version 1) available at Research Square.

[8] Jiyun, Z., Hongpeng, W. & Z., Z. Cnnh_pss: protein 8-class secondary structure prediction by convolutional neural network with highway. BMC bioinformatics 19, 99–109, (2018).

[9] Lina, Y., Pu, W., Zhong, X, L. & Y., T. Y. Protein structure prediction based on bn-gru method. Int. J. Wavelets, Multiresolution Inf. Process. 18, 2050045, (2020).

[10] L, W. A. & L., M. S. Reasons for the occurrence of the twenty coded protein amino acids. J. Mol. Evol. 17, 273–284, (1981).

[11] Ashraf, Y. & Li, Y. Template-based c8-scorpion: a protein 8-state secondary structure prediction method using structural information and context-based features. BMC bioinformatics 15, 1–8, (2014).

[12] Zhao, Y. & Liu, Y. Oclstm: Optimized convolutional and long short-term memory neural network model for protein secondary structure prediction. Plos one 16, e0245982, DOI: https://doi.org/10.1371/journal.pone.0245982 (2021).

[13] Cuff, J. A. & Barton, G. J. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. Proteins: Struct. Funct. Bioinforma. 34, 508–519, (1999).

[14] Tomasz, S. & K., R.-K. I. S. Protein secondary structure prediction: a review of progress and directions. Curr. Bioinforma. 15, 90–107, (2020).

[15] F, A. S., M, G. E., R., A., A., S. A. & Yu, Y. K. Psi-blast pseudocounts and the minimum description length principle. Nucleic acids research 37, 815–824, (2009).

[16] F, A. S. et al. Gapped blast and psi-blast: a new generation of protein database search programs. Nucleic acids research 25, 3389–3402, (1997).

[17] Xiaoyang, J., Qiwen, D., D, H. & R., L. Amino acid encoding methods for protein sequences: a comprehensive review and assessment. IEEE/ACM transactions on computational biology bioinformatics 17, 1918–1931, (2019).

[18] Michael, H., A, E. & Wang, Y. Modeling aspects of the language of life through transfer-learning protein sequences. BMC bioinformatics 20, 1–17, (2019).

[19] Peng, W. S., Ma, J., J & J., X. Protein secondary structure prediction using deep convolutional neural fields. Sci. reports 6, 1–11, (2016).

[20] Zeming, L., J, L. & Y., Q. Must-cnn: a multilayer shift-and-stitch deep convolutional architecture for sequence-based protein structure prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, (2016).

[21] Rosenberg, J. A., Sønderby, C. K., Sønderby, S. K. & Winther, O. Deep recurrent conditional random field network for protein secondary prediction. In Proceedings of the 8th ACM international conference on bioinformatics, computational biology, and health informatics, 73–78, (2017).

[22] Michael, S. V., Z., S. & A., A. Predicting secondary structure of protein using hybrid of convolutional neural network and support vector machine. Int. J. Intell. Eng. & Syst. 14, (2022).