Mathematical Biosciences and Engineering

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

MBE, 20(2): 2203–2218.
DOI: 10.3934/mbe.2023102
Received: 29 September 2022
Revised: 04 November 2022
Accepted: 13 November 2022
Published: 17 November 2022

Research article

Protein secondary structure prediction based on Wasserstein generative adversarial networks and temporal convolutional networks with convolutional block attention modules

Lu Yuan, Yuming Ma* and Yihui Liu*

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

* **Correspondence:** Email: mym@qlu.edu.cn, yxl@qlu.edu.cn.

Abstract: As an important task in bioinformatics, protein secondary structure prediction (PSSP) is not only beneficial to protein function research and tertiary structure prediction, but also to promote the design and development of new drugs. However, current PSSP methods cannot sufficiently extract effective features. In this study, we propose a novel deep learning model WGACSTCN, which combines Wasserstein generative adversarial network with gradient penalty (WGAN-GP), convolutional block attention module (CBAM) and temporal convolutional network (TCN) for 3-state and 8-state PSSP. In the proposed model, the mutual game of generator and discriminator in WGAN-GP module can effectively extract protein features, and our CBAM-TCN local extraction module can capture key deep local interactions in protein sequences segmented by sliding window technique, and the CBAM-TCN long-range extraction module can further capture the key deep long-range interactions in sequences. We evaluate the performance of the proposed model on seven benchmark datasets. Experimental results show that our model exhibits better prediction performance compared to the four state-of-the-art models. The proposed model has strong feature extraction ability, which can extract important information more comprehensively.

Keywords: protein secondary structure prediction; Wasserstein generative adversarial network; convolutional block attention module; temporal convolutional network; feature extraction

1. Introduction

Protein is an organic macromolecule composed of amino acids, which is the key prerequisite for all life activities. Protein primary structure is the sequence of amino acids in a peptide chain [1]. Secondary structure is a specific structure formed by the atoms of the polypeptide backbone coiled or folded along a certain axis. On the basis of the secondary structure, a tertiary structure can be further formed. Secondary structure is a bridge connecting primary structure and tertiary structure, so it is a key step in the study of tertiary structure [2,3]. Protein secondary structure prediction (PSSP) not only helps us to further understand the structure and function of proteins but also can facilitate the design and development of drugs. Since experimental methods for PSSP are expensive and time-consuming, we use deep learning methods for PSSP.

Generally, secondary structure can be divided into 3 states or 8 states. The 3-state secondary structure is divided into helix (H), strand (B), and coil (C) [4,5]. The 8-state secondary structure is divided into α-helix (H), helix (G), π-helix (I), β-bridge (B), β-sheet (E), bend (S), turn (T), and coil (C) [6]. Compared with the 3-state secondary structure, the 8-state secondary structure information is more abundant, which is more beneficial to protein research, but the prediction of the 8-state is more complex and challenging.

Early research work using machine learning methods such as *k*-nearest neighbors [7], support vector machines [8] and neural networks [9] mainly focused on 3-state PSSP. In recent years, neural networks with deep architectures have been widely used in various fields including PSSP and achieved outstanding results. The SSpro utilized BRNN and structural similarity for PSSP [10]. The GSN can learn Markov chains to sample from conditional distributions and be used for PSSP [11]. The SPIDER3 server combined LSTM and BRNN to extract long-range interactions in protein sequences [12]. The SSREDNs used a deep encoder-decoder model to learn the sequence-structure relationship of proteins [13]. The combination of CNN and RNN can extract local and long-range dependencies in protein sequences, which achieves remarkable performance in PSSP [2,14]. The SAINT utilized the deep 3I network and self-attention mechanism to predict secondary structure [15].

Inspired by the effectiveness and development of deep learning techniques in PSSP, we propose a novel deep learning model WGACSTCN for PSSP using Wasserstein generative adversarial network with gradient penalty (WGAN-GP) [18], convolutional block attention module (CBAM) [19] and temporal convolutional network (TCN) [20]. We use PSSM profile and one-hot encoding as input to the model. In our model, WGAN-GP can efficiently extract protein features, and TCN with CBAM (CBAM-TCN) can capture key deep local and long-range dependencies in residue sequences. To evaluate the model, we conduct extensive experiments on the public test sets CASP10, CASP11, CASP12, CASP13, CASP14, and CB513.

The main contributions of this study include: (1) We propose CBAM-TCN by introducing CBAM in TCN, which can extract key features in protein sequences. (2) We use the sliding-window technique and zero-filling method to enable the sequence-to-sequence network CBAM-TCN to process sequence-to-label data so that local features can be extracted. (3) We propose a novel deep learning method WGACSTCN combining WGAN-GP, local CBAM-TCN module and long-range CBAM-TCN module. (4) The results show that the proposed model exhibits better performance in 3-state and 8-state PSSP compared to other popular models.

2. Materials and methods

2.1. Datasets

The CullPDB [21] dataset consists of 14,991 protein chains selected from the Protein Data Bank by the PISCES [21] server, where the percent identity cutoff is set to 25%, the resolution cutoff is set to 3 angstroms, and the R-factor cutoff is set to 0.25. PISCES is widely used in PSSP, which can produce lists of sequences according to mutual sequence identity and chain-specific criteria. This study uses the classification rules of the DSSP [8] program to ensure correct secondary structure information. We deleted the proteins in the CullPDB dataset that duplicated the test set and also deleted proteins with lengths less than 40 or more than 800. The final CullPDB contains 14,562 proteins. To better evaluate the model, our dataset is divided into: a training set (11,650), a validation set (1456) and a test set (1456). We use six public test sets CASP10 [22], CASP11 [23], CASP12 [24], CASP13 [25], CASP14 [26], and CB513 [27] to further evaluate the model performance, where all CASP datasets are from the website https://predictioncenter.org/. The test sets CASP10, CASP11, CASP12, CASP13, CASP14, and CB513 contain 99, 81, 19, 22, 23, and 513 proteins, respectively.

2.2. Feature representation

In this study, we use two feature representation methods: one-hot amino acid encoding and position-specific scoring matrix (PSSM) [28]. Protein sequences consist of 20 standard amino acid types and 6 non-standard amino acid types. Usually, the 6 non-standard amino acid types are considered as a single type, so the protein sequence is considered to be composed of 21 amino acid types. An amino acid sequence of length L can be represented as an $L \times 21$ feature vector through one-hot encoding, where the corresponding component of the amino acid is 1, and the remaining components are 0. Since each type of vector in one-hot encoding is orthogonal to each other, it can be called orthogonal encoding.

PSSM has rich biological evolution information, which is widely used in PSSP. PSSM can represent the amino acid sequence as an $L \times 20$ scoring sequence by aligning the sequence itself with multiple sequences, where 20 represents the 20 standard amino acid types. PSI-BLAST [29] can generate PSSM based on a threshold of 0.001 and 3 iterations.

2.3. Evaluation criteria

For evaluation, we use four metrics in this paper: Q3 accuracy, Q8 accuracy and segment overlap (SOV) [30] score for 3-state and 8-state PSSP.

The eight classes of secondary structure are H, G, I, E, B, S, T, and C. The three types of secondary structure are H, B, and C. Q3 and Q8 accuracy are defined as follows:

$$Q3 = \frac{S_C + S_E + S_H}{S} \times 100 \tag{1}$$

$$Q8 = \frac{S_H + S_G + S_I + S_E + S_B + S_C + S_T + S_S}{S} \times 100 \tag{2}$$

where S represents the number of all amino acids and S_i ($i \in \{H, E, C\}$ or $\{H, G, I, E, B, C, T, S\}$)

represents the correct number of predicted secondary structure type *i*.

Let the number of all residues in the sequence be N_{SOV}, all observed structural segments be S_1, all predicted segments be S_2, and all overlapping segments between S_1 and S_2 be S_0. The SOV score is calculated as:

$$SOV = \frac{100}{N_{SOV}} \sum_{S_0} \left[\frac{minov(S_1, S_2) + \sigma(S_1, S_2)}{maxov(S_1, S_2)} length(S_1)\right] \tag{3}$$

where $maxov(S_1, S_2)$ represents the length of the union of segments S_1 and S_2, and $minov(S_1, S_2)$ represents the length of the intersection of segments S_1 and S_2. $\sigma(S_1, S_2)$ allows changes at the observed segment boundaries and is calculated as:

$$\sigma(S_1, S_2) = \min \begin{Bmatrix} maxov(S_1, S_2) - minov(S_1, S_2) \\ minov(S_1, S_2) \\ int[len(S_1)]/2 \\ int[len(S_2)]/2 \end{Bmatrix} \tag{4}$$

2.4. Wasserstein GAN with gradient penalty (WGAN-GP)

The generative adversarial network (GAN) [16] exhibits powerful image denoising and feature extraction capabilities in complex distributed data, but it has the problems of difficulty in training and slow convergence. The Wasserstein GAN (WGAN) [17] improves the stability of training, but it still suffers from problems such as vanishing gradients. Therefore, we use WGAN with gradient penalty (WGAN-GP) for feature extraction of protein sequences.

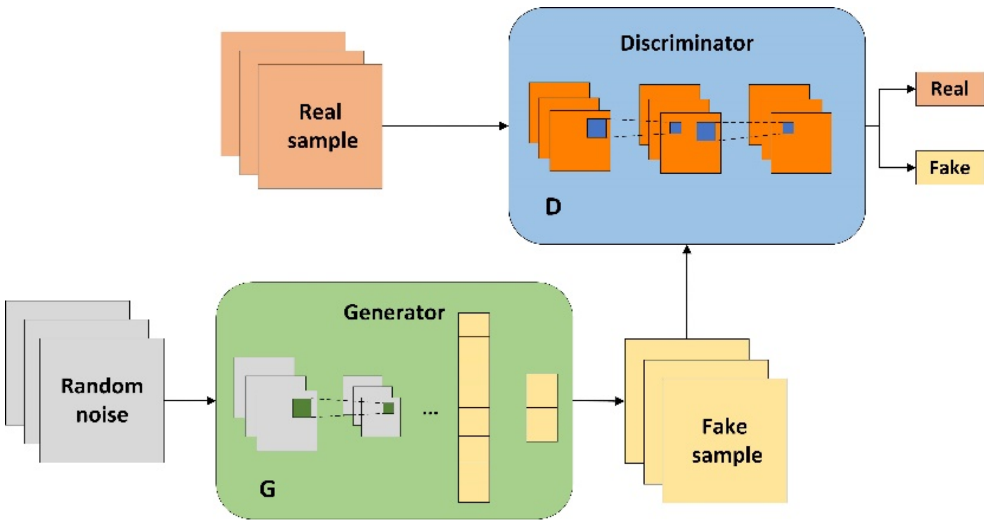


Figure 1. The WGAN-GP model architecture.

As shown in Figure 1, WGAN-GP consists of G and D, where G is the generator and D is the discriminator. The G can transform random noise into data similar to amino acid sequences by simulating the complex distribution relationship of real samples, but the generated data is not real. The D can learn the difference between real and fake data to distinguish whether the data comes from the generator or the real sample. The mutual game between the G and the D can continuously optimize the performance of the network. The loss of WGAN-GP is calculated as follows:

$$L=\underbrace{\underset{\tilde{x}\sim P_g}{E}[D(\tilde{x})]-\underset{x\sim P_r}{E}[D(x)]}_{\text{Original critic loss}}+\underbrace{\lambda\underset{\hat{x}\sim P_{\hat{x}}}{E}[(\|\nabla_{\hat{x}}D(\hat{x})\|_2-1)^2]}_{\text{Gradient penalty}}\cdot \tag{5}$$

where x represents the real sample, $\tilde{x}$ represents the generated fake sample, P_g represents the generated sample distribution, P_r represents the real sample distribution, and λ represents the hyperparameter with a default value of 10. The $P_{\hat{x}}$ distribution is a random sampling distribution after P_r and P_g are sampled once each.

The WGAN-GP utilizes gradient penalty to satisfy the 1-Lipschitz condition while preventing drastic changes in weights, which can enhance the learning ability and stability of the network.

2.5. Temporal convolutional networks (TCN)

The TCN proposed in recent years not only achieves superior performance in various fields but also outperforms popular networks such as recurrent neural network in most aspects. Furthermore, the TCN model has fast computation speed, low memory occupation, and stable training.

2.5.1. Causal convolutions

Two characteristics of TCN are: 1) The model can map inputs of any length to inputs of the same length. 2) Information cannot leak from the future to the past. To achieve the first characteristic, the TCN uses a 1D fully convolutional network, where zero padding is used to ensure that all layers have the same length. To satisfy the second characteristic, the TCN uses causal convolution. As shown in Figure 2(a), the output at time t in causal convolution is only affected by the input at previous time and time t.

2.5.2. Dilated convolutions

To capture very long-term effective history, as shown in Figure 2(b), TCN uses dilated convolutions to expand the receptive field. The dilated convolution operation F on the sequence element s is calculated as follows:

$$F(s)=(X*{}_d f)(s)=\sum_{i=0}^{k-1}f(i)\cdot X_{s-d\cdot i} \tag{6}$$

where X is the input, f and k are the filter and the size of the filter, $s-d\bullet i$ is the direction of the history, and d is the dilation factor. The d increases exponentially with the network depth, ($d=2^i$ at layer i), so the dilated convolution at the first layer is equivalent to the regular convolution. The top-level output of the network can represent a wider range of inputs as the number of layers increases.

2.5.3. Residual connections

As shown in Figure 2(c), TCN uses residual connections to avoid the problem of training instability as the model deepens. The output of the block is calculated as follows:

$$o = Activation(X + F(x)) \tag{7}$$

The basic structure of the residual block is shown in Figure 2(d). The dilation factors of the two convolutional layers are the same. A weight normalization layer, a ReLU layer, and a dropout layer are added after each dilated causal convolutional layer to enhance speed and stability during training. In all residual blocks of TCN, the input is added to the output. When the width of the input and output of a block is different, we can perform 1 × 1 convolution on the input to ensure the addition.

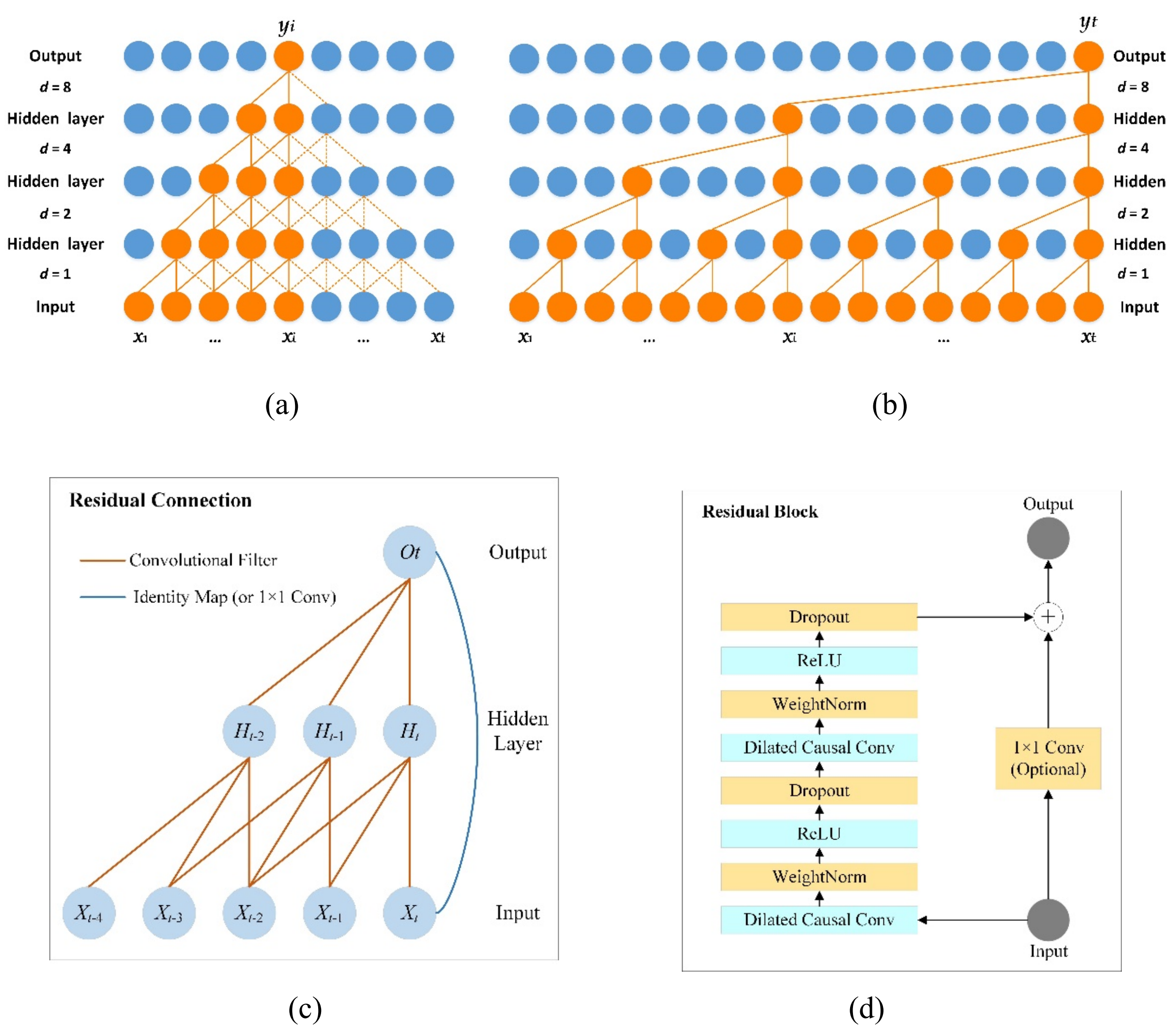
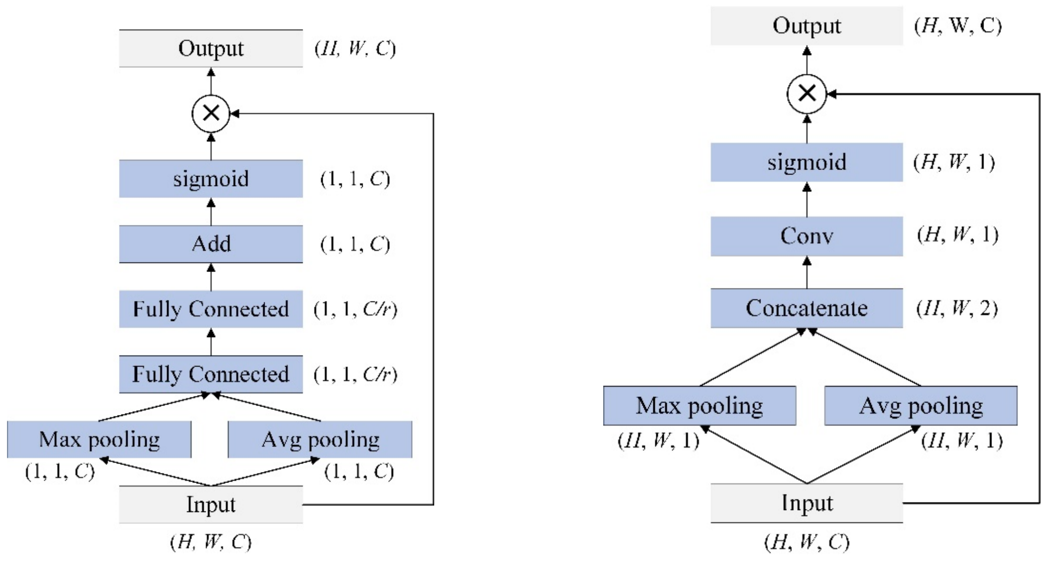


Figure 2. Structures in TCN. (a) Causal convolutions. (b) Dilated convolutions. An example using four residual blocks. (c) TCN residual connection. (d) TCN residual block. An example using two convolutional layers.

2.6. CBAM-TCN

2.6.1. Channel attention



(a) The channel attention structure (b) The spatial attention structure

Figure 3. Attention mechanism in CBAM.

Figure 3(a) shows the detailed structure of the channel attention. The input feature map $x_c(i, j) \in R^{H \times W \times C}$ is first subjected to global max pooling and global average pooling operations. Two fully connected layers are used to share parameters, where r is the reduction rate. Finally, we merge the two feature maps and use the sigmoid to get weight coefficients of size $1 \times 1 \times C$. Global average pooling X_{ac} and global max pooling X_{mc} in channel attention are calculated as follows:

$$X_{ac} = \text{ReLU}\left(\frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} x_c(i, j), 0\right) \tag{8}$$

$$X_{mc} = \max\left[\text{ReLU}\left(x_c(i, j)\right), 0\right], i \in \{1, \cdots, H\}, j \in \{1, \cdots, W\} \tag{9}$$

2.6.2. Spatial attention

The spatial attention structure is shown in Figure 3(b), which can obtain spatial weights. First, we perform max-pooling and average-pooling on the input $x_s \in R^{H \times W \times C}$ to get two feature maps. The two feature maps are concatenated in the channel dimension. Then, we use convolutional layers to reduce the channels and use the sigmoid function to generate spatial weight coefficients of size $H \times W \times 1$. In spatial attention, average pooling and max pooling are calculated as follows:

$$F_{as} = \text{ReLU}\left(\frac{1}{k} \sum_{i=1}^{k} z_s(i), 0\right) \tag{10}$$

$$F_{ms} = \max\left[\text{ReLU}\left(z_s(i)\right), 0\right], i \in \{1, \cdots, k\} \tag{11}$$

where F_{as} and F_{ms} are the spatial average pooling weight and max pooling weight generated at the s-th position of the output, respectively, z_s is the vector form of x_s, and s is equal to $H \times W$.

2.6.3. The proposed CBAM-TCN

The CBAM consists of channel and spatial attention modules. CBAM sequentially uses channel attention and spatial attention to obtain refined features, which can extract key information in different channels and spaces. Compared with optimization algorithms [31–33] such as prairie dog optimization algorithm [34], Gazelle optimization algorithm [35], and Aquila optimizer [36], CBAM focusing on channels and spaces has strong generalization and low memory.

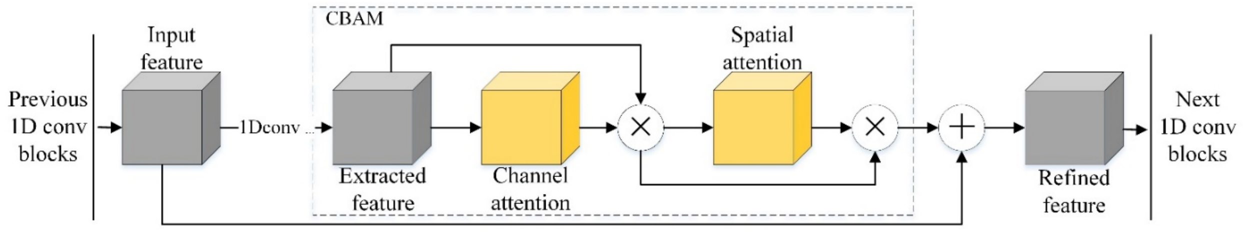


Figure 4. The CBAM-TCN residual block structure. The input features are from the previous residual block. CBAM is used to optimize features extracted by a series of 1D convolutions.

The feature information of different channels or spaces in the protein sequence has different effects on the recognition of secondary structure, so we propose the CBAM-TCN model, which combines CBAM and TCN to capture key feature information in amino acid sequences. The detailed structure of the block in CBAM-TCN is shown in Figure 4. We add the CBAM after a series of convolution operations in the residual block to further obtain the attention weight information of the extracted features.

2.7. The proposed WGACSTCN

The proposed method mainly contains four modules: input, WGAN-GP module, CBAM-TCN local feature extraction module, CBAM-TCN long-range feature extraction module and output. The architecture of our method is shown in Figure 5.

The input part mainly includes feature generation and data processing. First, we transformed the initial protein data into PSSM features of size $20 \times L$ and one-hot features of size $21 \times L$, where L is the sequence length. Therefore, the input to the model is the hybrid features of size $41 \times L$. Since the secondary structure is mainly influenced by the local residues in the amino acid sequence, we can segment the sequence into feature matrices of size $41 \times W$ by the sliding window technique, where W is the size of the window and each feature matrix corresponds to a single label.

In the WGAN-GP module, the mutual game between the discriminator and the generator can fully extract the features in protein sequences. We use three layers of 2D convolutions in the generator to generate protein sequences and three layers of 2D convolutions in the discriminator to discriminate between real and fake protein samples.

In the CBAM-TCN local feature extraction module, we use four residual blocks with the attention mechanism to capture the key deep local interactions between amino acid residues. We use three layers of dilated causal convolutions with the same dilation factor in the residual block. After each layer of dilated causal convolution, we also use an instance normalization layer to accelerate the model convergence, a ReLU layer to ensure the stability of the gradient and a spatial dropout layer to prevent the risk of network overfitting. In addition, since CAMB-TCN is a sequence-to-sequence prediction network, when using the sliding window technique, we set the position W of the label sequence as the secondary structure label corresponding to the segmented short sequence, and set the remaining positions to 0.

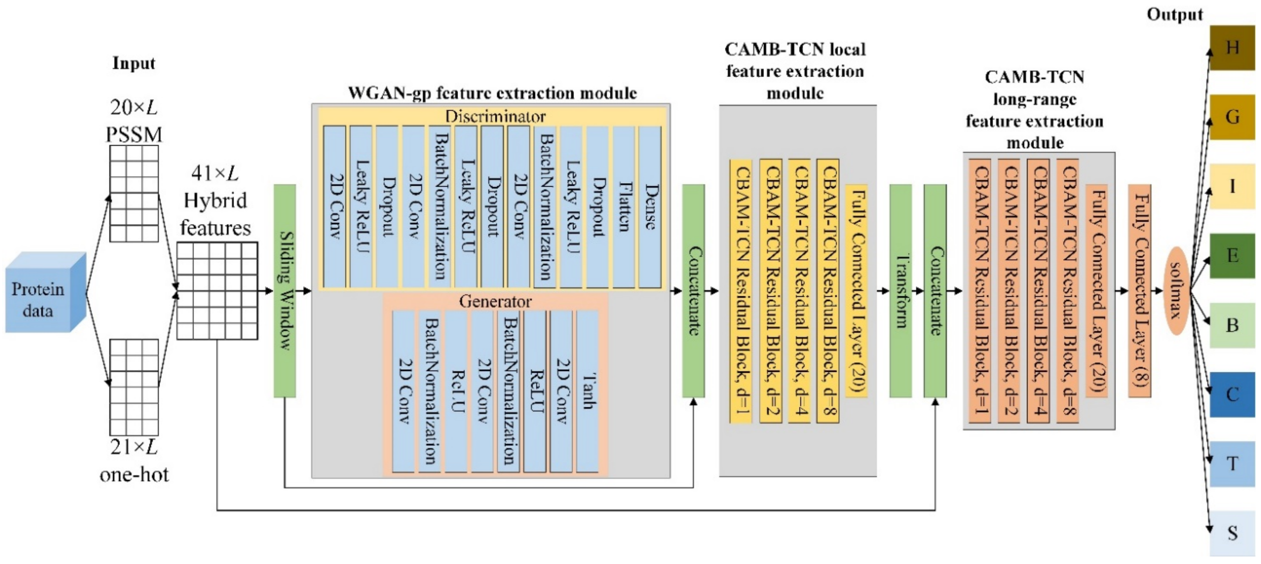

Input
20×L PSSM
21×L one-hot
Protein data
41×L Hybrid features
Sliding Window
WGAN-gp feature extraction module
Discriminator
2D Conv
Leaky ReLU
Dropout
2D Conv
BatchNormalization
Leaky ReLU
Dropout
2D Conv
BatchNormalization
Leaky ReLU
Dropout
Flatten
Dense
Generator
2D Conv
BatchNormalization
ReLU
2D Conv
BatchNormalization
ReLU
2D Conv
Tanh
Concatenate
CAMB-TCN local feature extraction module
CBAM-TCN Residual Block, d=1
CBAM-TCN Residual Block, d=2
CBAM-TCN Residual Block, d=4
CBAM-TCN Residual Block, d=8
Fully Connected Layer (20)
Transform
Concatenate
CAMB-TCN long-range feature extraction module
CBAM-TCN Residual Block, d=1
CBAM-TCN Residual Block, d=2
CBAM-TCN Residual Block, d=4
CBAM-TCN Residual Block, d=8
Fully Connected Layer (20)
Fully Connected Layer (8)
softmax
Output
H
G
I
E
B
C
T
S

Figure 5. The detailed structure of the proposed WGACSTCN.

In the CBAM-TCN long-range feature extraction module, we first concatenate the input features and the extracted local features into matrices of size $61 \times L$. We then use four CBAM-TCN residual blocks to further capture key deep long-range dependencies in residue sequences. The broad receptive field of residual blocks composed of 1D convolutional architectures enables flexible long-range interactions of elements between layers. Therefore, we still use three dilated causal convolutional layers in each residual block to extract features.

In the output part, we use the softmax function to complete the classification of the secondary structure.

The proposed model can use more comprehensive amino acid information for prediction to improve the problem of insufficient feature extraction. Furthermore, the proposed method can utilize key local and long-range interactions in residue sequences to facilitate connections between input features and secondary structures, thereby better modeling complex sequence-structure mapping relationships.

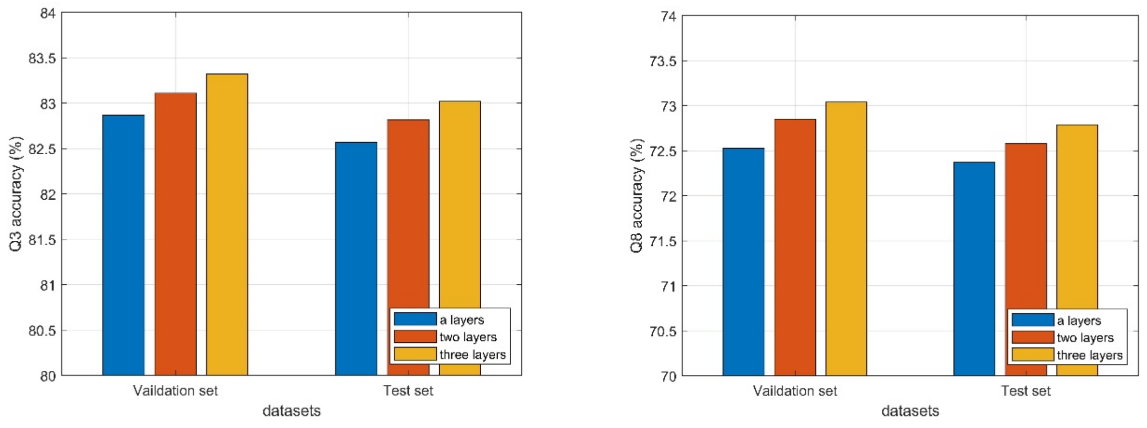
3. Results

3.1. The performance of the proposed model

In this section, we show the effect of different parameters on PSSP in three modules of the model. To make our model show good performance in PSSP, we conduct extensive experiments on the CullPDB dataset, where the evaluation metrics are Q3 accuracy, Q8 accuracy and SOV score.

3.1.1. Impact of WGAN-GP module parameters

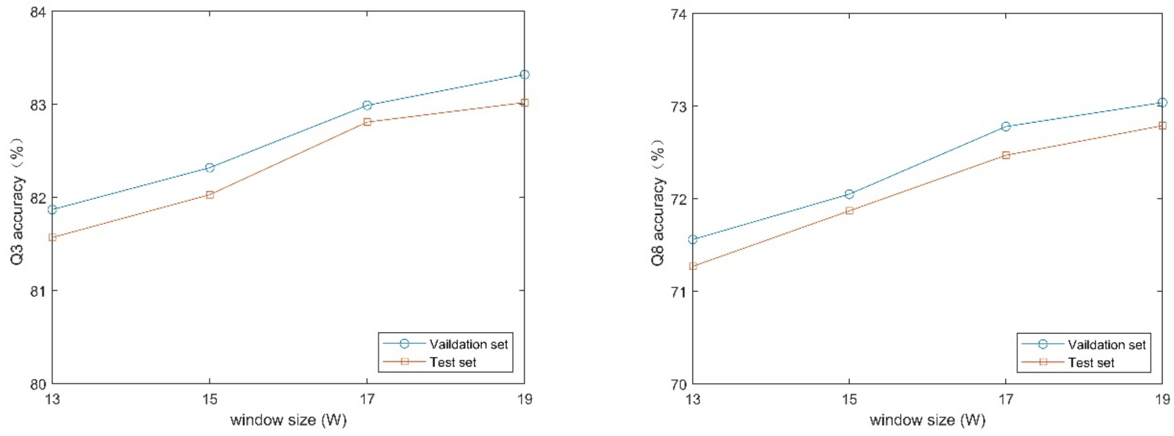
To explore the effect of the number of convolutional layers in the WGAN-GP module on the proposed model, we conduct comparative experiments on validation and test sets using 1, 2, and 3 convolutional layers, respectively. As shown in Figure 6, the model achieves the best experimental results in 3-state and 8-state PSSP when using 3 convolutional layers. Because the discriminator and generator cannot perform better when the number of layers is too small. Furthermore, too many convolutional layers can also make the discriminator and generator unbalanced.



(a) Q3 accuracy (b) Q8 accuracy

Figure 6. Model performance under different number of convolutional layers on the CullPDB dataset.

3.1.2 Impact of CBAM-TCN local feature extraction module parameters

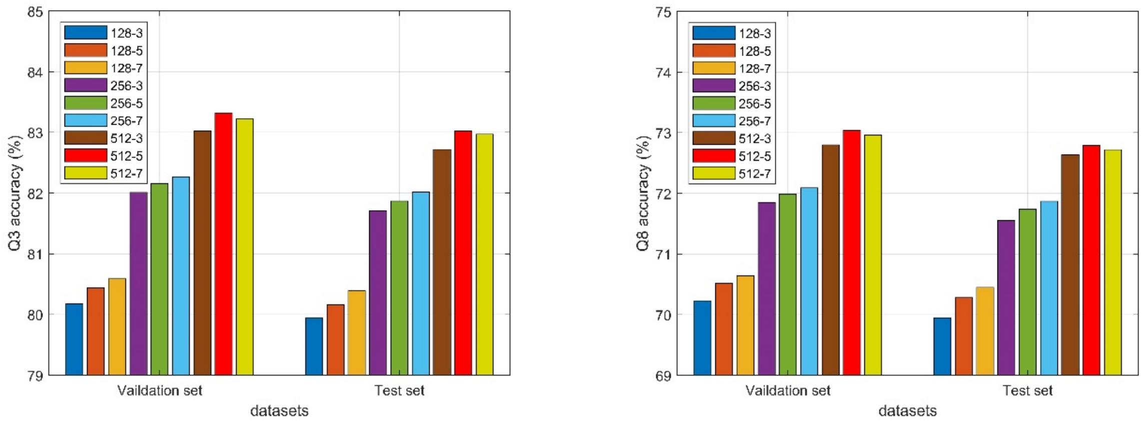


(a) Q3 accuracy (b) Q8 accuracy

Figure 7. Model performance under different sliding window sizes on the CullPDB dataset.

Secondary structure recognition is mainly affected by residues on a local scale, so we performed comparative experiments using protein sequences segmented with window sizes of 13, 15, 17, and 19, respectively. Figure 7 shows the Q3 and Q8 accuracy of our model on the validation and test sets. The model achieves the best classification performance on the 2 datasets when the sliding window size is 19. Because a large window can contain a wider range of protein information, but too large a window may cause key information to be missed.

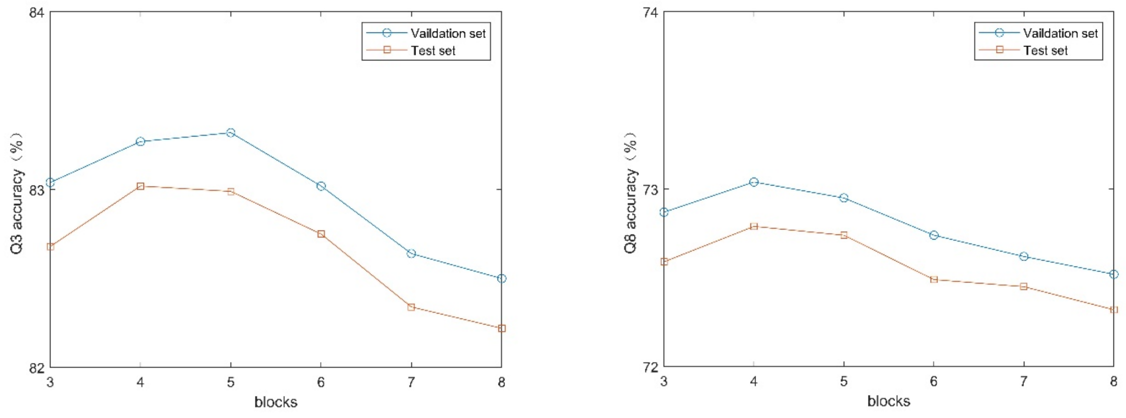
To verify the effect of the size and number of filters on secondary structure prediction, as shown in Figure 8, we conduct comparative experiments on the validation and test sets. The proposed model achieves the highest Q3 and Q8 accuracy when the size and number of filters are 5 and 512. Because the size and number of filters determine the range and channels of feature information extraction, which affect the capture of key information in the sequence.



(a) Q3 accuracy (b) Q8 accuracy

Figure 8. Model performance under different numbers and sizes of filters on the CullPDB dataset.

3.1.3. Impact of CBAM-TCN long-range feature extraction module parameters



(a) Q3 accuracy (b) Q8 accuracy

Figure 9. Model performance under different numbers of CBAM-TCN residual blocks on the CullPDB dataset.

The CAMB-TCN residual blocks directly control the depth of our model, so we use different CBAM-TCN residual block numbers 3, 4, 5, 6, 7, and 8 for comparison experiments respectively.

The 3-state and 8-state PSSP accuracy of the model on the validation and test sets are shown in Figure 9. The Q8 accuracy is the highest on the two datasets when the model uses 4 residual blocks, while the Q3 accuracy is the maximum on the two datasets when the number of residual blocks is 4 and 5, respectively. The main reason is that too few residual blocks of the model will make it difficult to extract deeper features, and too many residual blocks of the model will increase the computational complexity and the risk of overfitting. Therefore, we use 4 CBAM-TCN residual blocks to capture key deep interactions in residue sequences with filter sizes and numbers of 7 and 512.

3.2. Comparison with popular models

In this section, we compare our model with four state-of-the-art models DCRNN [37], CNN_BIGRU [38], DeepACLSTM [39], MUFOLD-SS [40] on seven datasets CullPDB, CASP10, CASP11, CASP12, CASP13, CASP14, and CB513. DCRNN utilizes convolutional neural networks with different filter sizes to capture local interactions and a recurrent neural network composed of gated units to extract long-range interactions in sequences. CNN_BIGRU utilizes convolutional neural networks and bidirectional gated recurrent units for prediction. DeepACLSTM uses an asymmetric convolutional neural network and a bidirectional long short-term memory network to extract local and global dependencies in residue sequences. MUFOLD-SS predicts secondary structure based on the deep inception-inside-inception model. For the fairness of the comparison, all models are trained using our CullPDB dataset with hybrid features PSSM + one-hot as input.

Table 1. Comparison with popular models on seven datasets in 3-state PSSP.

Methods	CullPDB		CASP10		CASP11		CASP12		CASP13		CASP14		CB513	
	Q3	SOV	Q3	SOV	Q3	SOV	Q3	SOV	Q3	SOV	Q3	SOV	Q3	SOV
DCRNN	82.12	78.51	82.57	75.71	80.57	75.53	80.41	74.75	80.49	77.09	80.28	71.46	84.66	79.63
CNN_BIGRU	82.31	78.68	82.40	76.20	81.03	76.58	80.37	75.62	80.64	76.94	80.45	**71.92**	84.81	79.90
DeepACLSTM	82.64	79.45	83.43	77.76	81.32	76.04	80.49	75.56	80.91	77.43	80.79	71.73	85.02	80.12
MUFOLD-SS	**83.02**	**79.62**	83.28	78.04	81.68	77.41	80.94	77.47	81.15	78.02	**81.12**	70.97	**85.30**	**80.23**
WGACSTCN	82.94	79.23	**83.85**	**78.93**	**82.01**	**77.90**	**80.99**	**77.62**	**81.80**	**78.04**	81.02	71.36	84.97	80.01

Table 2. Comparison with popular models on seven datasets in 8-state PSSP.

Methods	CullPDB		CASP10		CASP11		CASP12		CASP13		CASP14		CB513	
	Q8	SOV	Q8	SOV	Q8	SOV	Q8	SOV	Q8	SOV	Q8	SOV	Q8	SOV
DCRNN	72.06	70.42	72.11	69.74	70.50	68.44	69.41	67.24	68.05	68.01	68.87	63.27	75.63	73.06
CNN_BIGRU	72.28	70.15	71.87	69.17	70.94	69.05	69.67	68.06	67.83	67.92	68.69	62.95	75.54	72.84
DeepACLSTM	72.86	71.34	73.09	71.42	71.24	**69.93**	69.82	68.18	68.47	69.31	69.52	63.46	76.01	73.46
MUFOLD-SS	**73.32**	**71.59**	72.98	71.51	**71.62**	69.84	70.23	69.32	68.23	68.89	69.23	62.69	**76.64**	**74.04**
WGACSTCN	72.79	71.08	**73.11**	**71.56**	71.02	69.81	**70.36**	**69.34**	**68.59**	**69.50**	**69.68**	**63.54**	75.72	73.17

To evaluate the performance of 3-state and 8-state PSSP, we use Q3 accuracy, Q8 accuracy and SOV score as overall evaluation measures. The results of comparing our method with four popular methods on seven datasets are shown in Tables 1 and 2. It can be seen that the Q3 accuracy, Q8 accuracy and SOV score of the proposed method outperform the four existing popular classifiers in most cases. This is mainly due to the powerful feature extraction capability of the proposed WGACSTCN, which can effectively extract protein information of residue sequences while capturing the key deep local and long-range dependencies in residue sequences, thus making full use of more comprehensive features to improve the performance of 3-state and 8-state PSSP. However, the wide receptive field in the proposed WGACSTCN may also lose some important information.

4. Conclusions

In this study, we propose a novel deep learning model WGACSTCN for sequence-to-sequence prediction based on WGAN-GP, CBAM, and TCN. We use PSSM profile and one-hot feature encoding as input to the model. To extract key feature information in protein sequences, we combine CBAM and TCN to propose a CBAM-TCN model. In the proposed model, we use the WGAN-GP module to extract the protein information of the sequence. Then, we use the CBAM-TCN local extraction module to capture deep local features among amino acid residues, where the residue sequences are segmented by a sliding window method. We also use the CBAM-TCN long-range extraction module to further capture deep long-range interactions in sequences. We test the performance of the proposed model on seven benchmark datasets using Q3 accuracy, Q8 accuracy and SOV score as evaluation metrics. Experimental results on 3-state and 8-state PSSP show that our model outperforms four existing popular models. Our model can effectively improve the problem of insufficient feature extraction through the successful combination of deep architecture modules. Furthermore, the excellent feature extraction capability of the WGACSTCN can more comprehensively capture longer-term key interactions in residue sequences to facilitate the accurate recognition of secondary structure. However, the proposed model can only extract information unidirectionally, so in the future we will improve the feature extraction of the model while enriching the input features to improve PSSP.

Acknowledgments

This work was supported by the National Natural Science Foundation of China [grant number 61375013] and the Natural Science Foundation of Shandong Province [grant number ZR2013FM020]. The codes and datasets for this work are at https://github.com/yuanlu9707/WGAN-CBAM-TCN.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. Y. Yang, J. Gao, J. Wang, R. Heffernan, J. Hanson, K. Paliwal, et al., Sixty-five years of the long march in protein secondary structure prediction: the final stretch, *Briefings Bioinf.*, **19** (2018), 482–494. https://doi.org/10.1093/bib/bbw129
2. P. Kumar, S. Bankapur, N. Patil, An enhanced protein secondary structure prediction using deep learning framework on hybrid profile based features, *Appl. Soft Comput.*, **86** (2020), 105926. https://doi.org/10.1016/j.asoc.2019.105926

3. G. Wang, Y. Zhao, D. Wang, A protein secondary structure prediction framework based on the extreme learning machine, *Neurocomputing*, **72** (2008), 262–268. https://doi.org/10.1016/j.neucom.2008.01.016
4. 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. https://doi.org/10.1186/1471-2105-15-S8-S3
5. 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. https://doi.org/10.1038/s41598-018-28084-8
6. 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. https://doi.org/10.1002/bip.360221211
7. S. Salzberg, S. Cost, Predicting protein secondary structure with a nearest-neighbor algorithm, *J. Mol. Biol.*, **227** (1992), 371–374. https://doi.org/10.1016/0022-2836(92)90892-N
8. M. H. Zangooei, S. Jalili, Pssp with dynamic weighted kernel fusion based on svm-phgs, *Knowl. Based Syst.*, **27** (2012), 424–442. https://doi.org/10.1016/j.knosys.2011.11.002
9. N. Qian, T. J. Sejnowski, Predicting the secondary structure of globular proteins using neural network models, *J. Mol. Biol.*, **202** (1988), 865–884. https://doi.org/10.1016/0022-2836(88)90564-5
10. C. N. Magnan, P. Baldi, Sspro/accpro 5: almost perfect prediction of protein secondary structure and relative solvent accessibility using profiles, machine learning and structural similarity, *Bioinformatics*, **30** (2014), 2592–2597. https://doi.org/10.1093/bioinformatics/btu352
11. J. Zhou, O. Troyanskaya, Deep supervised and convolutional generative stochastic network for protein secondary structure prediction, in *International Conference on Machine Learning*, PMLR, (2014), 745–753.
12. 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. https://doi.org/10.1093/bioinformatics/btx218
13. Y. Wang, H. Mao, Z. Yi, Protein secondary structure prediction by using deep learning method, *Knowl. Based Syst.*, **118** (2017), 115–123. https://doi.org/10.1016/j.knosys.2016.11.015
14. M. S. Klausen, M. C. Jespersen, H. Nielsen, K. K. Jensen, V. I. Jurtz, C. K. Soenderby, et al., Netsurfp-2.0: Improved prediction of protein structural features by integrated deep learning, *Proteins Struct. Funct. Bioinf.*, **87** (2019), 520–527. https://doi.org/10.1002/prot.25674
15. M. R. Uddin, S. Mahbub, M. S. Rahman, M. S. Bayzid, Saint: self-attention augmented inception-inside-inception network improves protein secondary structure prediction, *Bioinformatics*, **36** (2020), 4599–4608. https://doi.org/10.1093/bioinformatics/btaa531
16. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, et al., Generative adversarial nets, *Commun. ACM*, **63** (2020), 139–144. https://doi.org/10.1145/3422622
17. M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in *International Conference on Machine Learning*, PMLR, (2017), 214–223.
18. I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. C. Courville, Improved training of wasserstein gans, in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, (2017), 1–11.

19. S. Woo, J. Park, J.-Y. Lee, I. S. Kweon, Cbam: Convolutional block attention module, in *Proceedings of the European Conference on Computer Vision (ECCV)*, (2018), 3–19. https://doi.org/10.1007/978-3-030-01234-2_1
20. S. Bai, J. Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, preprint, arXiv:1803.01271.
21. G. Wang, R. L. Dunbrack, Pisces: recent improvements to a pdb sequence culling server, *Nucleic Acids Res.*, **33** (2005), W94–W98. https://doi.org/10.1093/nar/gki402
22. 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** (2014), 1–6. https://doi.org/10.1002/prot.24452
23. J. Moult, K. Fidelis, A. Kryshtafovych, T. Schwede, A. Tramontano, Critical assessment of methods of protein structure prediction: Progress and new directions in round xi, *Proteins Struct. Funct. Bioinf.*, **84** (2016), 4–14. https://doi.org/10.1002/prot.25064
24. 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** (2018), 7–15. https://doi.org/10.1002/prot.25415
25. A. Kryshtafovych, T. Schwede, M. Topf, K. Fidelis, J. Moult, Critical assessment of methods of protein structure prediction (casp)—round xiii, *Proteins Struct. Funct. Bioinf.*, **87** (2019), 1011–1020. https://doi.org/10.1002/prot.25823
26. A. Kryshtafovych, T. Schwede, M. Topf, K. Fidelis, J. Moult, Critical assessment of methods of protein structure prediction (casp)—round xiv, *Proteins Struct. Funct. Bioinf.*, **89** (2021), 1607–1617. https://doi.org/10.1002/prot.26237
27. J. A. Cuff, G. J. Barton, Evaluation and improvement of multiple sequence methods for protein secondary structure prediction, *Proteins Struct. Funct. Bioinf.*, **34** (1999), 508–519. https://doi.org/10.1002/(SICI)1097-0134(19990301)34:4<508::AID-PROT10>3.0.CO;2-4
28. D. T. Jones, Protein secondary structure prediction based on position-specific scoring matrices, *J. Mol. Biol.*, **292** (1999), 195–202. https://doi.org/10.1006/jmbi.1999.3091
29. S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, et al., Gapped blast and psi-blast: a new generation of protein database search programs, *Nucleic Acids Res.*, **25** (1997), 3389–3402. https://doi.org/10.1093/nar/25.17.3389
30. A. Zemla, Č. Venclovas, K. Fidelis, B. Rost, A modified definition of sov, a segment-based measure for protein secondary structure prediction assessment, *Proteins Struct. Funct. Bioinf.*, **34** (1999), 220–223. https://doi.org/10.1002/(SICI)1097-0134(19990201)34:2<220::AID-PROT7>3.0.CO;2-K
31. L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, A. H. Gandomi, The arithmetic optimization algorithm, *Comput. Methods Appl. Mech. Eng.*, **376** (2021), 113609. https://doi.org/10.1016/j.cma.2020.113609
32. L. Abualigah, A. Diabat, P. Sumari, A. H. Gandomi, Applications, deployments, and integration of internet of drones (iod): A review, *IEEE Sens. J.*, **21** (2021) 25532–25546. https://doi.org/10.1109/JSEN.2021.3114266
33. L. Abualigah, M. Abd Elaziz, P. Sumari, Z. W. Geem, A. H. Gandomi, Reptile search algorithm (rsa): A nature-inspired meta-heuristic optimizer, *Exp. Syst. Appl.*, **191** (2022), 116158. https://doi.org/10.1016/j.eswa.2021.116158
34. A. E. Ezugwu, J. O. Agushaka, L. Abualigah, S. Mirjalili, A. H. Gandomi, Prairie dog optimization algorithm, *Neural Comput. Appl.*, **2022** (2022), 1–49. https://doi.org/10.1007/s00521-022-07530-9

35. J. O. Agushaka, A. E. Ezugwu, L. Abualigah, Gazelle optimization algorithm: A novel nature-inspired metaheuristic optimizer, *Neural Comput. Appl.*, **2022** (2022), 1–33. https://doi.org/10.1007/s00521-022-07854-6
36. L. Abualigah, D. Yousri, M. Abd Elaziz, A. A. Ewees, M. A. Al-Qaness, A. H. Gandomi, Aquila optimizer: a novel meta-heuristic optimization algorithm, *Comput. Ind. Eng.*, **157** (2021), 107250. https://doi.org/10.1016/j.cie.2021.107250
37. Z. Li, Y. Yu, Protein secondary structure prediction using cascaded convolutional and recurrent neural networks, preprint, arXiv:1604.07176.
38. 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, preprint, arXiv:1811.07143.
39. 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. https://doi.org/10.1186/s12859-018-2565-8
40. C. Fang, Y. Shang, D. Xu, Mufold-ss: New deep inception-inside-inception networks for protein secondary structure prediction, *Proteins Struct. Funct. Bioinf.*, **86** (2018), 592–598. https://doi.org/10.1002/prot.25487