Knowledge-Based Systems 237 (2022) 107771

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Protein secondary structure prediction using a lightweight convolutional network and label distribution aware margin loss

Wei Yang [a], Zhentao Hu [b], Lin Zhou [b], Yong Jin [b,*]

[a] Henan Key Laboratory of Big Data Analysis and Processing, School of Computer and Information Engineering, Henan University, Kaifeng, 475004, China
[b] School of Artificial Intelligence, Henan University, Zhengzhou, 450046, China

ARTICLE INFO

Article history:
Received 30 December 2020
Received in revised form 10 November 2021
Accepted 12 November 2021
Available online 1 December 2021

Keywords:
Protein secondary structure prediction
Imbalanced classification
Deep convolutional network
Lightweight network
Channel shuffle

ABSTRACT

Protein secondary structure prediction (PSSP) is an important task in computational molecular biology. Recently, deep neural networks have demonstrated great potential in improving the performance of eight-class PSSP. However, the existing deep predictors usually have higher model complexity and ignore the class imbalance of eight-class secondary structure data in training. In addition, the current methods cannot guarantee that the features corresponding to the padded residue positions are always zero during the forward propagation of the network, which will cause the prediction results of the same protein chain to be different under varied zero-padding numbers. To this end, we propose a novel lightweight convolutional network ShuffleNet_SS, which adopts modified 1-dimensional batch normalization to eliminate the impact of padded residue positions on nonpadded residue positions and uses the label distribution aware margin loss to enhance the network's ability to learn rare classes. In particular, in order to enable ShuffleNet_SS to fully achieve cross-group information exchange, we further improve the standard channel shuffle operation. Experimental results on the benchmark datasets including CASP10, CASP11, CASP12, CASP13, CASP14 and CB513 show that the proposed method achieves state-of-the-art performance with much lower parameters compared to the five existing deep predictors.

1. Introduction

Predicting the secondary structure of a protein from its primary sequence is an important task in computational molecular biology. Proteins are versatile biological macromolecules composed of 20 amino acid types and play vital roles in many biological processes. Many applications such as the design of drugs and enzymes require knowledge of the structure of a protein in order to determine its function. Experimental methods such as X-ray crystallography, NMR spectroscopy and electron microscopy used to determine the three-dimensional structure of proteins are still extremely expensive and time-consuming. The number of protein structures currently released by the Protein Data Bank is still less than 200,000. However, due to the rapid development of protein sequencing technology, the number of known protein sequences in the UniParc database has reached more than 300 million. Therefore, to narrow the gap between the number of known protein sequences and the number of known protein structures, computationally predicting protein structures is becoming increasingly urgent. Protein secondary structure prediction (PSSP) is a crucial intermediate step for predicting protein tertiary structure [1]. Accurately predicted protein secondary structures can be used not only to predict protein structural classes [2], carbohydrate-binding sites [3], protein domains [4] and frameshifting indels [5] but also to construct many structural and functional analysis tools [6,7].

A protein secondary structure is a local folded structure determined by the interaction of a hydrogen bond donor and acceptor residues on a polypeptide's backbone atoms. Based on two main hydrogen-bonding patterns, "*n*-turns" and "bridges", Kabsch and Sander [8] divided secondary structures into eight categories: α-helix (H), 3_{10}-helix (G), π-helix (I), isolated β-bridge (B), extended strand (E), bend (S), hydrogen bonded turn (T) and loop or other irregular structures (L). Extended strand (E), which can appear in the form of parallel or antiparallel bridges, is determined by the long-range interaction between the residues. The remaining seven classes are mainly determined by the local interaction of residues. In particular, the limited flexibility of a peptide chain makes a secondary structure very sensitive to change in the residues on a polypeptide chain.

Given a protein chain composed of L amino acid residues, the goal of PSSP task is to assign a correct secondary structure

* Corresponding author.
E-mail address: jy@henu.edu.cn (Y. Jin).

https://doi.org/10.1016/j.knosys.2021.107771
0950-7051/

category to each amino acid residue. This prediction task can be further divided into two categories: three-state prediction and eight-state prediction. In three-state prediction, the eight secondary structure categories are reduced into three categories: helices (H, G, and I), strands (B and E), and coils (S, *T* and L). In particular, early works were mainly focused on three-state prediction due to the limited training data available. Many machine learning algorithms, such as support vector machines [9–11], neural networks [12–14], probabilistic graph models [15–17], and *k*-nearest neighbors [18,19], have been successfully used to perform this coarse-grained prediction. Compared with three-state prediction, eight-state prediction can provide more valuable local structural information [20] and is more challenging because eight states have an extremely imbalanced distribution in protein structures.

Inspired by the successful application of deep learning in many fields, such as image classification and natural language processing, the recently proposed PSSP methods are mainly use deep neural networks to perform eight-state prediction. SSpro8 [21] uses an ensemble of 100 bidirectional recursive neural networks for eight-state PSSP. Chou and Troyanskaya [22] introduce a convolutional architecture to the supervised generative stochastic network to perform prediction. In [23], a deep belief network based on restricted Boltzmann machines was proposed to predict secondary structures. The DeepCNF [24] utilizes an improved conditional neural field to model the complex sequence-structure mapping relationship and interdependency between adjacent secondary structure labels. The DCRNN [25] exploits multiscale convolutional neural network and stacked bidirectional gated recurrent units to extract local and global contexts. Subsequently, the literature [26] proposed a similar model called the CNN_BIGRU. In [27], a deep recurrent encoder–decoder network named the SSREDNs was proposed. DeepACLSTM [28] adopts asymmetric convolutional networks combined with bidirectional long short-term memory to predict secondary structure. In SPIDER3 [29], an iterative learning strategy is used to train the bidirectional recurrent neural network. Literature [30] incorporates the next-step conditioning technique into a multiscale convolutional neural network. MUFOLD-SS [31] exploits a new deep inception-inside-inception architecture to extract both local and nonlocal interactions between amino acids. Recently, SAINT [32] further introduced a self-attention mechanism for the deep inception-inside-inception network to improve MUFOLD-SS. Moreover, many other deep network variants have also been proposed to perform eight-state prediction [33–36].

When training a deep secondary structure prediction model, a specific number of protein chains need to be randomly selected from the training set to form a minibatch. In particular, the length of protein chains in minibatches is usually not equal due to the variable length distribution of protein chains. Therefore, it is necessary to perform zero padding on shorter protein chains so that all protein chains in the minibatch have the same length. However, the aforementioned deep secondary structure prediction networks will change the feature vectors corresponding to the padded positions to nonzero vectors during forward propagation, which in turn affects the prediction result of the nonpadded positions. Specifically, the same protein chain will obtain different prediction results under varied zero-padding numbers. Moreover, in the current deep prediction models, all secondary structure categories are treated equally during training without considering that their distribution is extremely imbalanced. Finally, existing deep secondary structure predictors usually have larger model sizes and higher computational costs.

To address these issues, we propose a novel eight-state secondary structure prediction model ShuffleNet_SS based on a lightweight convolutional network. We first modify the standard 1-dimensional batch normalization by introducing a mask matrix so that the features corresponding to the padding positions do not participate in the normalization operation and make their corresponding output features all zeros. Then, we design a basic network module by adopting modified batch normalization, depthwise separable convolution and channel shuffle. ShuffleNet_SS is constructed by stacking the basic network modules and thus has feature reuse ability. In particular, we find that the standard channel shuffle operation does not fully achieve cross-group information exchange in the stacked case. To this end, we propose an improved version that addresses this problem. Furthermore, considering that the eight-state secondary structures have an imbalanced class distribution, we adopt the label distribution aware margin loss [37] that encourages rare classes to have larger margins to train the proposed network. This can enhance the network's ability to learn rare classes without sacrificing the network's ability to fit frequent classes. Experimental results on several benchmark datasets show that ShuffleNet_SS achieves state-of-the-art secondary structure prediction performance with considerably fewer parameters.

2. The proposed deep convolutional network

In this section, we introduce a novel deep convolutional network architecture for sequence-to-sequence PSSP. We know that in order to train a secondary structure neural network model that performs sequence-to-sequence prediction, it is necessary to select a fixed number of protein chains from a randomly shuffled training dataset to form a minibatch. Since the selected protein chains usually have different lengths, the right side of the shorter protein chains usually needs to be padded with multiple zero values until their length is equal to the length of the largest protein chain in the minibatch. In particular, in order to ensure that all the padded positions do not affect the feature generation of other nonpadded positions, the feature vector corresponding to the former should always be a zero vector during the forward propagation process. However, when the feature data pass through a batch normalization layer [38] or a convolutional layer with a kernel width greater than 1, the feature vectors corresponding to the padded positions may become nonzero vectors. To solve this problem, we modified the standard 1-dimensional batch normalization by introducing a binary mask matrix and designed a new network module based on the modified batch normalization. In the following, we will first describe the 1D batch normalization with a mask matrix, then introduce the designed network module, and finally give the overall network framework for PSSP.

2.1. The 1D batch normalization with a mask matrix

The input of our network model includes two parts: a feature data tensor and a binary mask matrix. The shape of the feature data tensor is (N, C, L), where N represents the number of protein chains in a minibatch, C represents the length of the feature vector corresponding to each amino acid residue, and L represents the maximum protein chain length in a minibatch. The size of the binary mask matrix is (N, L), where 1 indicates a nonpadded position and 0 indicates a padded position. For given input feature data for PSSP, standard 1-dimensional batch normalization will calculate the mean and variance based on the features of the padded position and the nonpadded position at the same time. However, the feature vector corresponding to the padded position is an invalid numeric vector and should not be used to calculate the minbatch statistics. To this end, we allow 1-dimensional batch normalization to also receive a binary mask matrix as input. Let $\mathbf{X}$ be the 3D feature data tensor of our normalization layer and $\mathbf{M}$ be its corresponding mask matrix. Then, the mean and variance can be calculated as follows:

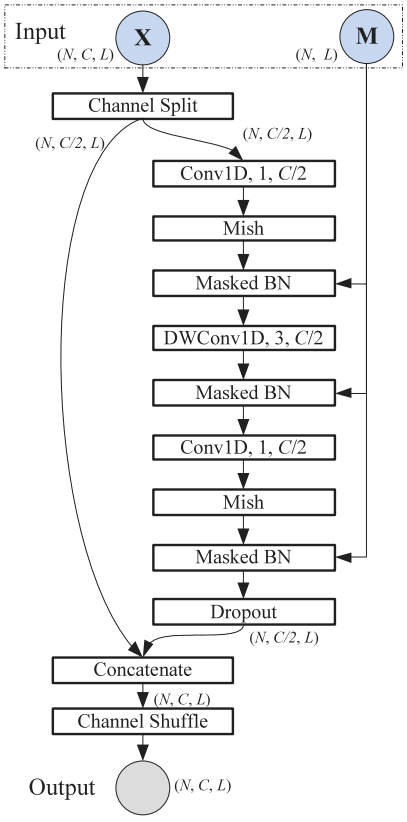

Fig. 1. Diagram of the designed network module. Conv1D:1D convolutional layer. DWConv1D:depthwise 1D convolution layer.

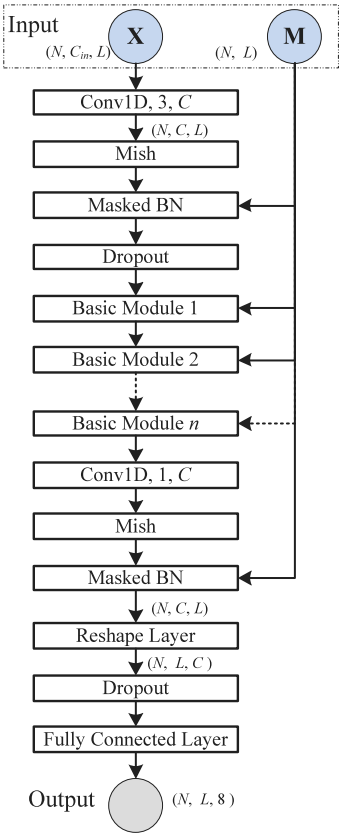

Fig. 2. The proposed network architecture for PSSP.

$$\mu_j = \sum_{i=1}^{N}\sum_{k=1}^{L} M_{i,k}X_{i,j,k} / \sum_{i=1}^{N}\sum_{k=1}^{L} M_{i,k}\ j = 1, \ldots, C \tag{1}$$

and

$$var_j = \sum_{i=1}^{N}\sum_{k=1}^{L} M_{i,k}(X_{i,j,k} - \mu_j)^2 / (\sum_{i=1}^{N}\sum_{k=1}^{L} M_{i,k} - 1)\ j = 1, \ldots, C \tag{2}$$

Note that the computation of the mean and variance is only based on the feature data of nonpadded positions indicated by the mask matrix. After calculating the mean $\boldsymbol{\mu} \in \mathbb{R}^{1\times C\times 1}$ and the unbiased variance $\mathbf{var} \in \mathbb{R}^{1\times C\times 1}$, as with standard batch normalization, we normalize the input feature tensor by first subtracting $\boldsymbol{\mu}$, and then dividing by $\sqrt{\mathbf{var} + \epsilon}$, where ϵ is a small positive constant for numerical stability. In addition, the calculation of the running mean and variance is the same as standard batch normalization. For the normalized feature data, affine transformation with two learning parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ is further performed. It should be noted that the transformation operation may convert the feature vector corresponding to the padded position into a nonzero vector. Therefore, we set the transformation output vectors corresponding to all padded positions to 0 vectors. This allows the output of the modified batch normalization to be directly used in any downstream network unit.

2.2. The designed network module

Our designed network module is shown in Fig. 1. The input of the module includes two parts: a 3D feature tensor **X** and a mask matrix **M**. It should be noted that the mask matrix **M** is only used to perform batch normalization to reduce internal covariate shift and speed up the network training. Here, we use "masked BN" to denote 1D batch normalization with a mask matrix. For the input feature tensor **X**, the channel split operator is used to split it into two branches evenly along the channel dimension. One branch remains unchanged for the purpose of feature reuse. The other branch uses the depthwise separable 1D convolution to reduce the computation costs and model size. The depthwise separable convolution usually consists of a depthwise convolution and a pointwise convolution [39]. The depthwise convolution keeps each channel separate and convolves each channel with the respective filter. Its output is a stack of all the convolved outputs. Obviously, the depthwise convolution cannot capture cross-channel dependencies. Therefore, the pointwise convolution, i.e., a 1D convolution (Conv1D) with a kernel size of 1, is then used to generate cross-channel features by combining the output of the depthwise convolution. In our network module, we use depthwise convolutions with a kernel size of 3. It should be noted that each side of the inputs of the depthwise convolution is zero padded by one residue to keep the size of the third dimension of the feature map fixed. In particular, in order to minimize memory access costs, the three convolutions in the module all use the same number of input and output channels according to the network structure design guidelines given in [40]. Considering that the Mish function $f(x) = x\tanh(\ln(1+e^x))$ [41] has better properties than the ReLU, we use it as the activation function. Furthermore, dropout is also introduced to the branch to prevent network overfitting. The outputs of the two branches are concatenated together along the channel dimension. As in [40], we introduce the channel shuffle operation [42] after concatenation to enable the information in the two branches to be communicated in the downstream module.

2.3. The overall network architecture

Building on the designed network module, we present the overall network architecture for eight-state PSSP in Fig. 2. In

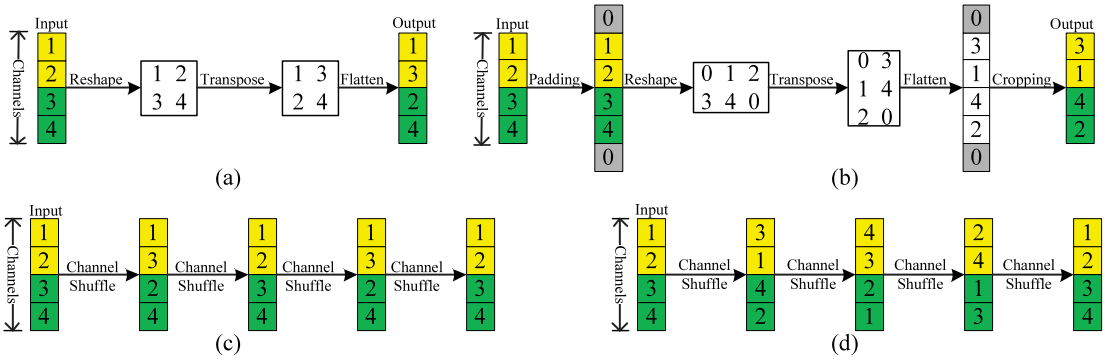

Fig. 3. Channel shuffle with two groups. All input channels are equally divided into two groups, which are marked with yellow and green. The digit indicates the number of the channel. In particular, we use 0 to denote the padded channel. (a) The standard channel shuffle. (b) The improved channel shuffle. (c) The input is processed four times continuously by the standard channel shuffle. The first and last channels of the input are fixed at their initial positions and cannot participate in the information exchange between the two groups. (d) The input is processed four times continuously by the improved channel shuffle. All channels of the input can be moved between two groups.

the proposed network, the first layer is a 1D full convolution with a kernel size of 3, which is followed by a Mish activation function, masked BN and dropout. It should be noted that the channel size has changed from C_{in} to C after performing the 1D full convolution. Then, n basic network modules in the style of Fig. 1 are repeatedly stacked to capture nonlocal interactions between residues. The output of the nth module is further fed to the 1D convolution with a kernel size of 1 to mix up features. The final fully connected layer is used as a classifier to output a classification score vector for each residue.

Note that each basic module divides the input feature tensor into two groups equally along the channel dimension. For basic modules stacked in the network, the output of the previous basic module after the channel shuffle operation is fed as the input to the next basic module. The purpose of introducing channel shuffle is to enable information to flow between two channel groups. However, we found that the standard channel shuffle operation [42] cannot fully achieve cross-group information exchange in the stacked case. Fig. 3a shows the operating steps of the standard channel shuffle. Given an input with $2 \times m$ channels, the standard channel shuffle operation first reshapes it into $(2, m)$ by transposing and then flattening it back to its original form. If only the single-step channel shuffle operation is considered, it can indeed achieve cross-group information exchange very well. However, in the stacked case as shown in Fig. 3c, the first and last channels of an input will remain unchanged and cannot participate in cross-group communication. In particular, for our network, this means that the content of the first channel of the output of the nth basic module is the same as that of the first channel of the input of the first basic module, which will prevent the network from generating new features during forward propagation and thus weaken the network's representation capability.

To solve the above problem, we improve the standard channel shuffle by introducing two additional steps of padding and cropping. For a given input, the padding step is first performed to fill an additional channel on both sides of the input along the channel dimension, then the padded input is processed by using the standard channel shuffle, and finally the two padded channels are removed by the cropping step. Fig. 3b shows the operating steps of the improved channel shuffle. The figure shows that the improved channel shuffle not only enables two groups to exchange channel information but can also change the positions of all channels in the input in a single step. Therefore, when it is used in the stacked case, it can truly achieve cross-group information exchange, as shown in Fig. 3d.

We name the proposed network ShuffleNet_SS. In ShuffleNet_SS, we exploit the depthwise separable convolution to reduce the model size and computational complexity. In particular, each depthwise convolution only requires $3 \times C/2$ parameters, which allows the network to increase the receptive field with fewer parameters. For each basic network module, half of the input features will remain fixed and participate in the computation of the downstream modules or units. This not only enables the network to have feature reuse ability, but also makes the final outputs have a diverse receptive field size. The maximum receptive field size of the network is $2 \times n + 3$. Moreover, most of the parameters and computation of ShuffleNet_SS are placed on the 1D convolutions with a kernel size of 1, which is significantly different from the existing PSSP network.

3. The loss function

The label distribution of eight-state protein secondary structure data has a heavy class-imbalance. In particular, the number of α-helix, the most frequent class, is usually more than 1000 times the number of π-helix, the rarest class. Therefore, when the standard cross-entropy loss without considering class imbalance is used for training, the rare classes can only obtain extremely low classification accuracy due to being overwhelmed by larger classes during the training process.

To alleviate the above problem, we use label distribution aware margin (LDAM) loss [37] to train the proposed PSSP network. Let n_k denote the number of training residues in the kth class and the vector $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_8]$, where $\gamma_k = 1/n_k^{1/4}$. The label-dependent margin of the kth class is defined as follows: $\Delta_k = \gamma_k/\gamma_{max}$, where γ_{max} is the maximum value of the vector $\boldsymbol{\gamma}$. Note that rare classes have a larger margin. Suppose the predicted logit tensor from the network is $\mathbf{Z} \in \mathbb{R}^{N \times L \times 8}$, and its corresponding target label and mask matrices are $\mathbf{Y}$ and $\mathbf{M}$, respectively. Then, the LDAM loss for each minibatch can be written as:

$$-\frac{1}{\sum_{i=1}^{N}\sum_{j=1}^{L} M_{ij}} \sum_{i=1}^{N}\sum_{j=1}^{L} M_{ij} \log \frac{e^{Z_{ijY_{ij}} - m\Delta_{Y_{ij}}}}{e^{Z_{ijY_{ij}} - m\Delta_{Y_{ij}}} + \sum_{k \neq Y_{ij}} e^{Z_{ijk}}} \tag{3}$$

where m is a hyperparameter to adjust the margin and is set as 1.0 in the experiments. The loss function in Eq. (3) can enhance the network's ability to learn rare classes without sacrificing the network's ability to fit frequent classes. In addition, it should be noted that different from the standard cross-entropy loss, the LDAM loss requires the use of a normalized fully connected layer as a classifier. The normalized fully connected layer calculates the logit score as:

$$\mathbf{z} = \tau \hat{\mathbf{W}}^T \hat{\mathbf{x}}, \ \ with\ \hat{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|} \ \ and\ \hat{\mathbf{W}}_{:j} = \frac{\mathbf{W}_{:j}}{\|\mathbf{W}_{:j}\|}, \ j \in \{1, \ldots, 8\} \tag{4}$$

where **x** is the feature mapping of a residue, **W** is the weight matrix, $\|\cdot\|$ denotes the L_2 norm, and τ is a scaling factor and is set as 16 in this study. The bias term **b** is discarded due to its negligible impact on the classification performance. The normalized fully connected layer can rectify the imbalance of decision boundaries and thus improve the classification performance.

In fact, in addition to the LDAM loss, there are other recently introduced losses, such as the class-balanced loss [43], focal loss [44], seesaw loss [45], equalization loss [46] and logit adjustment loss [47], that can effectively address the imbalanced classification problem with a long-tail distribution. In particular, the application scenarios of these losses for classification problems usually assume that the training set is class imbalanced while the validation and test sets are class balanced. However, all datasets used in eight-state PSSP are class imbalanced. In practice, we observe that only the LDAM loss can improve the prediction performance of the protein secondary structure; thus, it is used as the loss function to train our network.

4. Experiments

In this section, we evaluate the effectiveness of the proposed method on PSSP. Specifically, we first introduce the datasets used in the experiment and feature representation of residues, then describe the implementation details, and finally report the experimental results under the different settings.

4.1. Datasets

The PISCES server[1] can produce lists of sequences from the Protein Data Bank (PDB) using chain-specific criteria and the mutual sequence identity [48], which is often used to evaluate protein structure prediction algorithms [22]. Therefore, we used the file cullpdb_pc25_res2.5_R1.0_d200416_chains13482.fasta published by the PISCES server to construct a large nonhomologous dataset. This file was generated with the following parameter settings: the maximum resolution was 2.5 Å, the maximum R-value was 1.0 and the maximum sequence percentage identity between any two protein sequences was 25%. Furthermore, the file contained the primary sequence information of the protein chains and the corresponding PDB ID and chain ID. To obtain the corresponding eight-class secondary structure information of these protein chains, we further downloaded the secondary structure assignment file ss.txt[2] generated from the experimental coordinates using the DSSP program [8]. According to the PDB ID and chain ID, the corresponding secondary structure sequence can be extracted from the structure file. In the extraction step, those protein chains that have no corresponding structural information in the file ss.txt will be deleted. Moreover, we also removed protein chains with more than 800 or less than 50 residues. The final nonhomologous dataset consists of 12,510 protein chains and is called the CB12510 dataset. For evaluation purposes, we further randomly divide it into three parts: a training set (11,486), a validation set (512) and a test set (512).

To evaluate the performance of the proposed methods, we also use six commonly used datasets CB513, CASP10, CASP11, CASP12, CASP13 and CASP14 as test sets. The CB513 dataset was constructed by Cuff and Barton [49] in 2000. The original CB513[3] contains 513 nonhomologous protein chains. Some of these protein chains (such as 1AOZB-1,1AOZB-2 and 1AOZB-3) actually have the same chain ID and different digital codes. With the update of the PDB database, the protein structures of some PDB IDs that appear in CB513 (such as 1FDX, 1CHB, 2GCR, 1GEP, 1WSY, 1KIN and 3BCL) are outdated and replaced by newly released structures. Beyond this, we found that the primary sequence information of some protein chains (such as 1DAR-3 and 1TABI-1) also changed. Therefore, it is inappropriate to still use the original primary sequences as the test data. In particular, considering that the released protein secondary structure file ss.txt no longer divides protein chains with the same chain ID into multiple digital chains due to the discontinuity of backbone coordinates, we merge the sequences with the same PDB ID and chain ID in CB513 into a single chain. In addition, for those sequences in CB513 whose PDB ID has an outdated structural status, we use its chain ID and the corresponding replacement PDB ID to extract its primary sequence and secondary structure from the secondary structure file. Through the above processing operations, the final dataset consists of 433 protein chains and is called CB433. The maximum protein sequence length is 874. Moreover, in order to construct the five datasets CASP10, CASP11, CASP12, CASP13 and CASP14, we first downloaded the PDB IDs of the prediction targets from the website https://predictioncenter.org/. Then, according to the downloaded PDB IDs, the corresponding protein chains were extracted from the file ss.txt. It should be noted that a single PDB ID can correspond to multiple protein chains, and different protein chains can have the same primary sequence. To this end, we further filtered the protein chains in each dataset to remove the sequences with a sequence identity greater than 25%. Finally, the numbers of protein chains in the five datasets CASP10, CASP11, CASP12, CASP13 and CASP14 were 92, 84, 47, 41 and 33, respectively.

In order to investigate the PSSP performance on protein sequences with low-quality profile features, we further constructed a new test dataset called BC40_MSA_30. To construct this dataset, we first downloaded the BC40 dataset from the website https://drug.ai.tencent.com/protein/bc40/download.html. The dataset BC40 contains 36,976 protein chains. In particular, the dataset BC40, which was produced by MMseqs2 at 40% sequence identity, is the sequence clustering data released by the website https://cdn.rcsb.org/resources/sequence/clusters/bc-40.out on July 28, 2020. Then, each chain in the BC40 dataset was subjected to a multiple sequence alignment (MSA) search against the Uniref50 database using PSI-BLAST. Finally, the dataset BC40 was further filtered by deleting protein chains with MSA counts greater than 30. The resulting dataset contains 2009 protein chains and is called BC40_MSA_30.

For the above eight test datasets, dataset CB12510 must remove its homology with each test set before it could be used as a training set. To this end, for each test dataset, we keep it unchanged and filter dataset CB12510 to remove sequences with $>$ 25% sequence identity with the test dataset. After filtering, the numbers of protein chains in dataset CB12510 corresponding to the eight test datasets CB513, CB433, CASP10, CASP11, CASP12, CASP13, CASP14 and BC40_MSA_30 were 12012, 11987, 12360, 12336, 12376, 12453, 12504 and 11756, respectively. For these processed datasets, 512 protein chains were randomly selected as a validation set, and the remaining protein chains were selected as a training set. Moreover, the distributions of the eight secondary structure states in the nine datasets CB12510, CB513, CB433, CASP10, CASP11, CASP12, CASP13, CASP14 and BC40_MSA_30 are listed in Table 1. As shown in the table, all datasets have obvious class imbalanced distributions, and the distributions on different datasets are different.

4.2. Feature representation

For a given protein with L amino acid residues, in order to predict its secondary structure based on its primary sequence, each

[1] http://dunbrack.fccc.edu/Guoli/PISCES_ChooseInputPage.php
[2] https://cdn.rcsb.org/etl/kabschSander/ss.txt.gz
[3] http://www.compbio.dundee.ac.uk/jpred/legacy/data/pred_res/513_set.html

Table 1
Distribution of eight states (L, B, E, G, I, H, S and T) in the used datasets.

Datasets	L	B	E	G	I	H	S	T
CB12510	0.2456	0.0099	0.2054	0.0361	0.0002	0.3241	0.0756	0.1031
CASP10	0.2531	0.01	0.2259	0.0334	0.0002	0.2734	0.0828	0.1212
CASP11	0.2539	0.0115	0.2473	0.034	0	0.2649	0.0836	0.1047
CASP12	0.2702	0.0077	0.1905	0.0289	0	0.3255	0.0824	0.0948
CASP13	0.2883	0.0105	0.1803	0.0302	0	0.3042	0.0909	0.0956
CASP14	0.2898	0.0143	0.1708	0.0308	0.0044	0.3258	0.0751	0.0890
CB513	0.2114	0.0138	0.2128	0.0368	0.0004	0.3087	0.0981	0.118
CB433	0.2303	0.0133	0.2076	0.0366	0.0003	0.3026	0.0948	0.1145
BC40_MSA_30	0.3207	0.0090	0.1712	0.0240	0.0001	0.3083	0.0815	0.0852

amino acid residue in the sequence needs to be encoded as a numeric vector. In this study, we consider four amino acid encoding methods: one-hot encoding, residue embedding, position-specific scoring matrix (PSSM) encoding and MTX encoding. In the protein sequence database, the primary sequence of a protein is composed of 20 standard amino acid types (A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W and Y) and 6 nonstandard amino acid types (B, J, O, U, X and Z). In particular, the 6 nonstandard amino acid types are usually combined into a single type due to their low frequency of occurrence. Therefore, it can be considered that the primary sequence consists of 21 amino acid types. To this end, one-hot encoding represents each amino acid type as a 21-dimensional numeric vector, in which only one component is assigned a value of 1, and the remaining components are set to 0. One-hot encoding is also called orthogonal encoding because the vector representations of any two amino acid types are orthogonal to each other.

Different from one-hot encoding, which uses a sparse vector to represent each residue type, residue embedding maps each amino acid residue type to a dense vector with a specified embedding dimension. Residue embedding can be easily implemented by using the embedding operation in deep learning frameworks such as PyTorch and TensorFlow. In particular, in order to compare residue embedding with one-hot encoding, we set the embedding dimension to 21. This means that residue embedding requires learning a 21×21 embedding matrix, which needs to be initialized randomly before learning. Through one-hot encoding or residue embedding, a protein sequence of length L can be expressed as an $L \times 21$ feature matrix.

The PSSM and MTX encodings are profile matrices derived from the multiple sequence alignments of a target sequence. Such encoding contains the evolution information of residues, so it is generally regarded as a highly informative feature representation of protein sequences. The PSSM and MTX profile matrices can be generated simultaneously by using the PSI-BLAST program. In our experiments, they were calculated by PSI-BLAST against the UniRef50 database with an E-value inclusion threshold of 0.001 and 2 iterations. The UniRef50 database is a clustered set of sequences from known protein sequences, and all the sequences have less than 50% sequence identity. Before using PSI-BLAST, we used the pfilt program from the psipred package [50] to filter the UniRef50 database to remove low-complexity regions, transmembrane regions, and coiled-coil segments. When running PSI-BLAST, the parameter out_pssm is used to specify the output file name of the binary checkpoint file. After 2 iterations, the chkparser program from the psipred package is used to extract the MTX matrix from the output checkpoint file. For a protein sequence of length L, the PSSM matrix and the MTX matrix are both $L \times 20$, where 20 denotes the 20 standard types of amino acids. In order to use them as inputs for our deep model, we transform each entry in the PSSM matrix into the range $(0, 1)$ using the sigmoid function and divide each element in the MTX matrix by 1000. After processing, the value range of the elements in the MTX matrix is $(-1, 1)$. Note that both PSSM and MTX profile matrices are derived from the original ASCII position-specific matrix, so there is no complementarity between them.

4.3. Implementation details

The proposed network is implemented and experimented on using the PyTorch deep learning framework [51]. Unless explicitly specified in the experiment, the dropout ratio, the number of channels, and the number of basic network modules in our work are set to 0.2, 384 and 10, respectively. The AdamW [52] optimizer with the default hyperparameter values (β_1=0.9, β_2=0.999, $\epsilon = 10^{-8}$) is used to update the parameters of the network model. The learning rate and weight decay are all set to 0.001. In each training iteration, we randomly select 32 protein chains to form a minibatch. Moreover, the regularization method early stopping is used to halt the training of the network model when the overall per residue accuracy on the held-out validation dataset no longer improves. The "patience" value of early stopping is set to 5. All experiments are performed on a single NVIDIA GeForce GTX TITAN X GPU with 12 GB memory. The source code, datasets and pretrained models have been made available at https://github.com/fengtuan/ShuffleNet_SS.

4.4. Ablation study

To demonstrate the effect of feature representation and three hyperparameters (i.e., the number of channels (C), the number of basic modules (n) and the dropout ratio (p)) and batch normalization (BN) on the performance of the proposed method, we perform a number of experiments on the CB12510 dataset using the standard cross-entropy loss. It should be noted that the cross-entropy loss requires the use of a standard fully connected layer to calculate the logit score.

Effect of feature representation. The feature representation of each protein is determined by amino acid encoding methods. In this section, we analyze the effect of two profile features (PSSM and MTX) and four hybrid features (PSSM+one-hot, MTX+one-hot, PSSM+embedding, and MTX+embedding) on the performance of PSSP. Note that the hybrid features are obtained by concatenating a sequence feature and a profile feature, the dimensions of which are 41. Fig. 4 shows the Q_8 accuracy of the proposed network under six different feature representations on the CB12510 dataset. The figure shows that the four hybrid features have significantly better performance than the two single features on the validation and test sets. This means that the profile feature and the sequence feature contain complementary information for prediction. In particular, for the two single features PSSM and MTX, the latter is obviously better than the former. The Q_8 accuracy of PSSM+embedding on the validation set and test set is 0.22% and 0.12% higher than that on PSSM+one-hot, respectively. In addition, MTX+one-hot performs better than MTX+embedding on the validation set while its Q_8 accuracy on the test set is slightly lower than that of MTX+embedding. Considering that the two hybrid features MTX+one-hot and MTX+embedding have similar classification performances, and that residue embedding requires additional learning of the embedding matrix, we use the hybrid feature MTX+one-hot as the input feature of the proposed network.

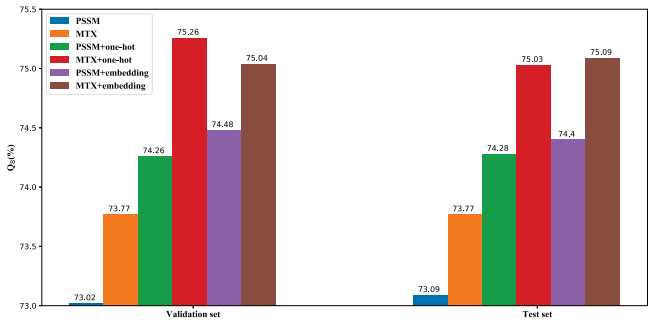

Fig. 4. The performance of the proposed network under different feature representations on the CB12510 dataset.

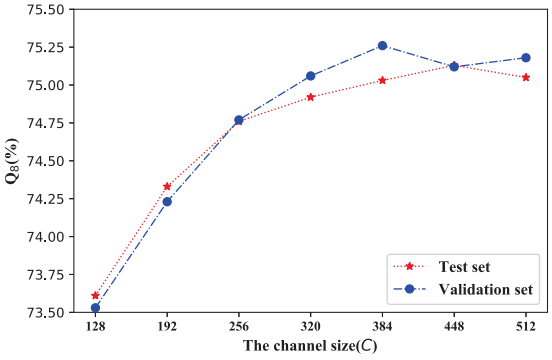

Fig. 5. The performance of the proposed network under different channel sizes on the CB12510 dataset.

Effect of the network width. To explore the effect of the network width on the performance of the proposed model, we conduct comparative experiments under different channel sizes 128, 192, 256, 320, 384, 448, 512. The Q_8 accuracy of our network on the validation and test sets is shown in Fig. 5. The figure shows that when the channel size is not greater than 384, the classification accuracy on the two datasets increases as the network width increases. In particular, when the network width is 384, the Q_8 accuracy on the validation set reaches the maximum. Therefore, we use 384 as the default width of the proposed network. Moreover, it should be noted that the network width not only affects the classification performance but also determines the model size and training time of the proposed network.

Effect of the network depth. The number of basic modules (n) determines the network depth of our network. We vary n from 7 to 14 and visualize the results in Fig. 6. It can be observed that the validation accuracy reaches the maximum at $n = 10$, while the test accuracy reaches the maximum at $n = 12$. In particular, a deeper network makes it easier for the network to fit the training data but increases the risk of overfitting.

Effect of the dropout ratio. Fig. 7 shows the Q_8 accuracy of the proposed network under different dropout ratios 0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3. The figure shows that when $p = 0$ (i.e., without using dropout), the proposed network has the worst classification performance on the validation and test sets. This suggests that although we have used batch normalization in the proposed network, dropout is still an effective regularization technique to overcome network overfitting for PSSP. Note that both validation accuracy and test accuracy reach the maximum at $p = 0.2$, so we use $p = 0.2$ as the default parameter of the proposed network.

Effect of batch normalization. In order to verify the effectiveness of using "masked BN" for PSSP, we compared it with the

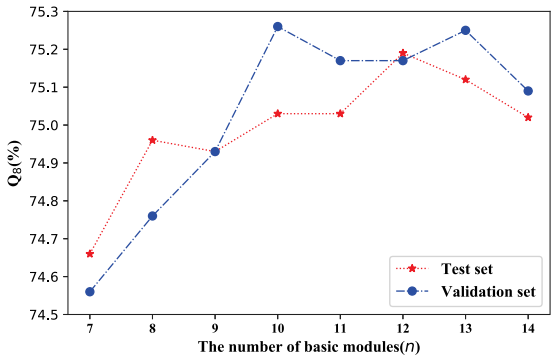

Fig. 6. The performance of the proposed network under different ns (ranging from 7 to 14) on the CB12510 dataset.

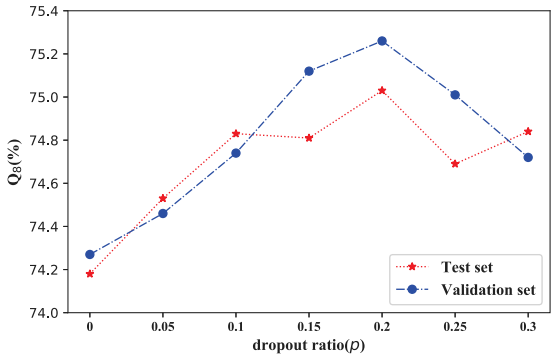

Fig. 7. The performance of the proposed network under different dropout ratios on the CB12510 dataset.

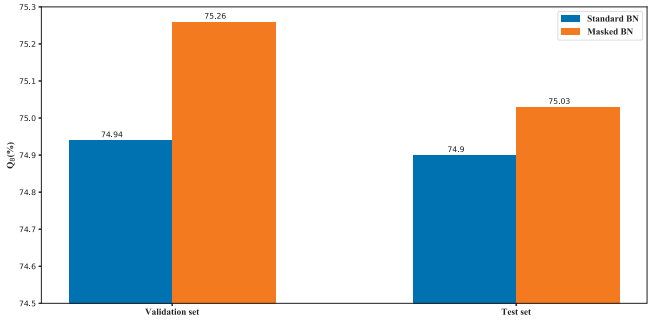

Fig. 8. The performance of the proposed network under different BNs on the CB12510 dataset.

standard BN. The comparison result is shown in Fig. 8. The figure shows that the masked BN consistently outperforms the standard BN on both the validation and test sets. This indicates that eliminating the impact of the padded positions on the nonpadded positions can improve the prediction performance of the eight-state secondary structure. Furthermore, it should be noted that when standard BN is used, the padded positions will affect the feature generation of the nonpadded positions during the forward propagation process. Hence, in the evaluation phase, standard BN will cause the same dataset to have slightly different prediction results under different batch sizes. This is because under different batch sizes, many protein sequences will have different zero-padding numbers. In particular, the change in the number of paddings will result in a very large change in the prediction results of the shorter sequences. Therefore, in order to avoid this inconsistency of prediction results, it is also necessary for PSSP to introduce masked BN.

Table 2
Comparison with state-of-the-art methods on the test set of CB12510. Bold indicates the best performance.

Methods	Q_L	Q_B	Q_E	Q_G	Q_I	Q_H	Q_S	Q_T	Q_8	F_1-score
CNN_BIGRU	72.17	0.97	**87.93**	27.21	0	**93.54**	17.66	50.73	73.56	44.98
DeepACLSTM	70.61	7.13	87.10	32.50	0	92.93	26.85	55.39	74.25	48.27
DCRNN	71.17	1.62	86.15	33.21	0	91.98	24.71	50.72	73.22	46.00
DeepCNN	**72.67**	6.65	83.82	30.89	0	93.36	27.91	53.86	74.09	47.94
MUFold-SS	**72.67**	6.32	87.25	29.78	0	93.42	28.49	55.70	75.00	48.67
ShuffleNet_SS(CE)	72.49	9.81	85.28	36.88	0	93.21	30.76	56.45	75.03	50.03
ShuffleNet_SS(LDAM)	70.12	**10.13**	86.92	**37.29**	0	93.07	**34.58**	**57.57**	**75.16**	**50.44**

4.5. Comparison with state-of-the-art methods

In this section, we compare the proposed methods with the five representative state-of-the-art methods CNN_BIGRU [26], DeepACLSTM [28], DCRNN [25], DeepCNN [30] and MUFold-SS [31] on the eight datasets, CB12510, CASP10, CASP11, CASP12, CASP13, CASP14, CB513 and CB433. Among the five methods, the first three algorithms are cascaded hybrid models of convolutional networks and bidirectional recurrent neural networks. The latter two algorithms are convolutional networks. The DeepCNN exploits a multiscale layer with convolution kernel sizes of 3, 7, and 9 for prediction; and MUFold-SS performs prediction based on a specially designed inception-inside-inception module. For a fair comparison, all methods also use the hybrid feature MTX+one-hot as the input. In particular, to investigate the influence of the network architecture on the classification performance, no additional techniques, such as the next-step condition [30], multitask learning [25] and ensemble learning, are used.

To evaluate the prediction performance of the eight-state protein secondary structure, we used two overall accuracy measures, Q_8 and the F_1-score; and eight single-state accuracy measures, Q_L, Q_B, Q_E, Q_G, Q_I, Q_H, Q_S, and Q_T. The comparison results on the eight datasets are given in Tables 2–9, respectively. In all these tables, "CE" is an abbreviation for the standard cross-entropy loss, which means that the corresponding ShuffleNet_SS uses the cross-entropy loss for training. The results show that ShuffleNet_SS outperforms the five existing algorithms in most cases in terms of the overall accuracy Q_8 and F_1-score. This is mainly because ShuffleNet_SS eliminates the impact of padded positions on nonpadded positions by using modified batch normalization and reduces the redundancy between connections through feature reuse, so it can better capture nonlocal interactions between residues. Although the recurrent neural network can capture long-range dependencies, the three hybrid models CNN_BIGRU, DeepACLSTM and DCRNN based on a recurrent neural network do not show any advantages over the three convolutional networks DeepCNN, MUFold-SS and ShuffleNet_SS. This may be because accurate secondary structure prediction requires a good balance between long-range interactions and local interactions. Moreover, it should be noted that the model size of ShuffleNet_SS is only 3.9 MB while the model sizes of CNN_BIGRU, DeepACLSTM, DCRNN, DeepCNN and MUFold-SS are 15.8 MB, 20.5 MB, 18.1 MB, 9.3 MB and 17.6 MB, respectively. Therefore, ShuffleNet_SS obtains state-of-the-art prediction performance with the fewest parameters.

In addition, note that the CB433 dataset is constructed by merging protein chains with the same PDB ID and replacing the PDB IDs with outdated information in the CB513 dataset. In particular, the two datasets contain approximately 300 identical protein chains. Comparing the results in Tables 8 and 9 shows that the performance on the CB433 dataset is better than that of the CB513 dataset in terms of the two overall accuracies in most cases. This indicates that the CB513 dataset is more difficult to predict because it contains incorrect information. Therefore, we propose replacing it with the CB433 dataset.

Compared with the cross-entropy loss, the LDAM loss achieved the highest F_1-score on all datasets except CASP14. On the CASP14 test set, the F_1-score of the LDAM loss is 0.99% lower than that of the CE loss while its Q_8 accuracy is 1.3% higher than that of the latter. These results demonstrate that the label distribution aware margin loss for imbalanced classification is an effective technique for improving eight-state PSSP. Note that on the test set of the CB12510 dataset, the LDAM loss not only improves the overall prediction accuracies of Q_8 and F_1-score but also improves the single-state prediction accuracy of the three rare classes B, G and S. However, on the other seven test sets, these three rare classes do not obtain similar improvements. This is mainly because for the test set of the CB12510 dataset, the distribution of the eight states in it is similar to the corresponding distribution on its training set. For the other seven datasets, these two distributions have different degrees of differences. In particular, the greater the difference between the two distributions, the more difficult the classification. Considering that there are differences between the class distributions on the training set and the test set and that they are both imbalanced, many recently proposed losses for long-tailed data, which usually assume that the class distribution on the training set is imbalanced and that the distribution on the test set is balanced, are not suitable to predict eight-class protein secondary structures. Therefore, the eight-state secondary structure prediction task is a good benchmark for testing imbalanced classification techniques.

Due to the extremely imbalanced frequency of occurrence of the eight-class secondary structures in the protein structure database, their single-state prediction accuracy differs greatly. The rare classes I and B are difficult to predict because they appear less frequently on the training set. As shown in the tables, the single-state accuracies Q_I of the rarest class I on all datasets are all 0% (note that π-helix (I) does not appear in the three datasets of CASP11, CASP12 and CASP13) while the frequent classes H and E have achieved very high prediction accuracy. In particular, the LDAM loss for imbalanced classification cannot improve Q_I. Moreover, Fig. 9 shows the confusion matrix obtained with ShuffleNet_SS (LDAM) on the CB433 dataset. Each entry of the confusion matrix indicates the percentage at which a given true category is predicted to be any category. The figure shows that the rarest class of the π-helix (I) is mainly incorrectly predicted as an α-helix (H), a hydrogen bonded turn (T) and an irregular structure (L). The second rarest class of the isolated β-bridge (B) is mainly incorrectly predicted as an irregular structure (L) and an extended strand (E). More efforts should be specifically made for these rare classes in the future.

4.6. Experimental results on the sequences with low-quality profile features

Protein chains with low sequence homology usually have limited prediction accuracy because their profile features obtained

Table 3
Comparison with state-of-the-art methods on the CASP10 dataset. Bold indicates the best performance.

Methods	Q_L	Q_B	Q_E	Q_G	Q_I	Q_H	Q_S	Q_T	Q_8	F_1-score
CNN_BIGRU	73.03	1.22	86.06	25.40	0	**94.01**	10.52	54.77	72.00	43.58
DeepACLSTM	72.65	9.39	**87.22**	32.72	0	92.83	24.63	57.86	73.71	48.68
DCRNN	73.34	2.86	84.84	31.50	0	93.91	21.24	53.26	72.70	46.19
DeepCNN	**75.25**	8.57	83.11	31.87	0	91.85	24.68	62.57	73.71	48.71
MUFold-SS	73.66	4.49	86.97	28.57	0	93.73	23.25	56.55	73.69	47.44
ShuffleNet_SS(CE)	72.12	10.20	87.11	32.60	0	93.65	**28.02**	**62.60**	74.64	49.98
ShuffleNet_SS(LDAM)	73.90	**11.84**	87.18	**37.00**	0	92.45	27.88	60.62	**74.68**	**50.44**

Table 4
Comparison with state-of-the-art methods on the CASP11 dataset. Bold indicates the best performance.

Methods	Q_L	Q_B	Q_E	Q_G	Q_I	Q_H	Q_S	Q_T	Q_8	F_1-score
CNN_BIGRU	**74.23**	1.15	81.39	24.74	0	92.05	17.41	48.99	70.81	49.73
DeepACLSTM	68.57	3.82	**84.36**	25.26	0	**93.01**	19.19	**53.56**	71.03	50.86
DCRNN	67.62	0.76	83.31	22.94	0	92.85	18.67	52.89	70.26	49.13
DeepCNN	70.57	2.29	81.73	27.96	0	90.43	25.17	53.18	70.74	51.15
MUFold-SS	70.52	3.05	83.35	25.13	0	92.17	22.13	53.10	71.23	51.45
ShuffleNet_SS(CE)	70.66	**4.58**	83.11	30.54	0	91.19	25.07	53.14	71.41	52.76
ShuffleNet_SS(LDAM)	69.79	4.20	83.66	**33.76**	0	92.00	**25.90**	52.14	**71.61**	**53.07**

Table 5
Comparison with state-of-the-art methods on the CASP12 dataset. Bold indicates the best performance.

Methods	Q_L	Q_B	Q_E	Q_G	Q_I	Q_H	Q_S	Q_T	Q_8	F_1-score
CNN_BIGRU	72.64	0.94	81.71	27.46	0	94.56	15.83	48.54	72.67	49.97
DeepACLSTM	68.08	3.77	83.77	25.94	0	93.48	22.28	47.08	71.85	50.47
DCRNN	69.54	3.77	**84.58**	30.48	0	93.57	19.10	47.92	72.38	50.91
DeepCNN	69.48	11.32	83.35	29.47	0	93.75	25.11	49.46	72.86	53.50
MUFold-SS	**73.12**	6.60	82.55	25.94	0	94.09	21.75	51.69	73.60	52.86
ShuffleNet_SS(CE)	70.40	13.21	82.05	38.04	0	**94.65**	22.19	**54.92**	**73.69**	55.21
ShuffleNet_SS(LDAM)	68.08	**15.09**	83.66	**39.29**	0	94.29	**25.73**	52.77	73.39	**56.08**

Table 6
Comparison with state-of-the-art methods on the CASP13 dataset. Bold indicates the best performance.

Methods	Q_L	Q_B	Q_E	Q_G	Q_I	Q_H	Q_S	Q_T	Q_8	F_1-score
CNN_BIGRU	**75.07**	0	76.62	12.20	0	92.17	3.87	47.69	68.78	50.17
DeepACLSTM	68.03	4.69	82.71	28.46	0	93.25	20.45	49.23	70.37	50.11
DCRNN	66.38	0	82.43	23.85	0	**94.57**	16.22	49.74	69.72	47.26
DeepCNN	65.19	**5.47**	79.66	28.18	0	93.33	20.45	**52.31**	69.32	49.61
MUFold-SS	70.64	0.78	80.94	17.07	0	93.17	13.78	48.46	69.71	46.81
ShuffleNet_SS(CE)	66.41	4.69	**84.16**	**29.27**	0	93.95	20.36	49.57	70.43	50.43
ShuffleNet_SS(LDAM)	67.52	4.69	83.57	28.18	0	93.60	**22.43**	48.97	**70.63**	**50.73**

Table 7
Comparison with state-of-the-art methods on the CASP14 dataset. Bold indicates the best performance.

Methods	Q_L	Q_B	Q_E	Q_G	Q_I	Q_H	Q_S	Q_T	Q_8	F_1-score
CNN_BIGRU	69.87	0.78	78.78	25.81	0	90.47	12.94	41.86	68.68	46.81
DeepACLSTM	68.73	2.33	79.11	23.30	0	88.60	11.91	40.99	67.59	45.58
DCRNN	70.25	0	76.26	28.67	0	**90.91**	12.21	39.25	68.29	45.97
DeepCNN	**70.75**	**3.88**	78.14	27.24	0	87.52	14.12	42.73	68.12	47.39
MUFold-SS	63.27	**3.88**	77.23	23.30	0	89.69	12.65	**47.83**	66.73	46.18
ShuffleNet_SS(CE)	65.56	**3.88**	79.30	**36.92**	0	89.15	**19.56**	44.10	68.17	**49.44**
ShuffleNet_SS(LDAM)	67.39	2.33	**82.92**	26.52	0	90.88	18.68	43.98	**69.47**	48.45

Table 8
Comparison with state-of-the-art methods on the CB513 dataset. Bold indicates the best performance.

Methods	Q_L	Q_B	Q_E	Q_G	Q_I	Q_H	Q_S	Q_T	Q_8	F_1-score
CNN_BIGRU	69.35	2.24	85.28	28.20	0	92.08	24.60	43.77	69.88	44.32
DeepACLSTM	70.41	2.75	83.62	26.32	0	92.17	18.20	46.25	69.38	43.31
DCRNN	**72.75**	1.03	82.36	27.39	0	92.38	17.56	47.09	69.72	43.26
DeepCNN	70.66	6.28	80.94	36.21	0	91.40	25.35	51.58	70.37	46.18
MUFold-SS	72.63	4.48	84.76	30.59	0	**92.50**	23.49	50.24	71.37	46.13
ShuffleNet_SS(CE)	70.44	6.45	84.14	**38.47**	0	92.16	29.17	52.33	71.79	47.74
ShuffleNet_SS(LDAM)	68.47	**7.57**	**86.18**	38.21	0	91.26	**29.51**	**54.93**	**71.87**	**48.20**

through multiple sequence alignments are low quality. In order to investigate the performance of state-of-the-art deep predictors on these protein chains, we further performed a comparative experiment on the BC40_MSA_30 dataset. In this experiment, it should be noted that the training and validation sets have high-quality profile features, while the test set has low-quality

Table 9
Comparison with state-of-the-art methods on the CB433 dataset. Bold indicates the best performance.

Methods	Q_L	Q_B	Q_E	Q_G	Q_I	Q_H	Q_S	Q_T	Q_8	F_1-score
CNN_BIGRU	70.00	1.49	84.12	22.74	0	**92.99**	16.90	53.42	70.29	43.38
DeepACLSTM	70.75	6.22	83.63	32.76	0	92.87	20.82	52.13	70.98	45.97
DCRNN	70.47	3.04	81.13	28.61	0	91.91	21.73	50.92	69.86	44.40
DeepCNN	**71.10**	5.73	80.54	30.70	0	90.54	27.56	54.23	70.51	46.15
MUFold-SS	70.75	4.46	83.97	31.21	0	92.25	23.38	53.62	71.20	46.07
ShuffleNet(CE)	70.81	7.21	83.06	**38.78**	0	92.48	26.74	**56.19**	72.02	47.98
ShuffleNet_SS(LDAM)	68.22	**7.28**	**85.22**	36.36	0	92.53	**31.27**	55.75	**72.17**	**48.34**

Table 10
Comparison of eight-state prediction results on the BC40_MSA_30 dataset. Bold indicates the best performance.

Methods	Q_L	Q_B	Q_E	Q_G	Q_I	Q_H	Q_S	Q_T	Q_8	F_1-score
CNN_BIGRU	61.99	0.12	73.60	12.32	0	**91.28**	5.56	29.14	63.85	38.84
DeepACLSTM	65.06	1.68	69.09	16.30	0	83.30	8.90	33.15	62.33	39.87
DCRNN	**66.88**	0.24	73.65	13.65	0	84.79	9.35	31.51	**63.97**	40.18
DeepCNN	62.13	1.16	73.11	15.54	0	85.97	11.43	**34.18**	63.17	40.39
MUFold-SS	64.21	0.32	67.57	12.50	0	86.93	9.47	33.60	62.90	39.38
ShuffleNet(CE)	64.84	2.00	73.62	15.04	0	84.37	10.02	32.96	63.41	40.63
ShuffleNet_SS(LDAM)	58.29	**2.08**	**75.40**	**16.46**	0	90.04	**13.04**	32.85	63.64	**41.33**

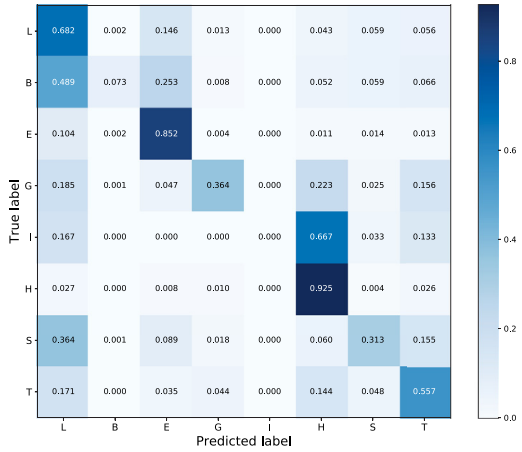

Fig. 9. Confusion matrix obtained with ShuffleNet_SS (LDAM) on the CB433 dataset.

Table 11
Comparison of three-state prediction results on the BC40_MSA_30 dataset. Bold indicates the best performance. The symbol "*" indicates that the accuracy data come from the published literature. The symbol "-" represents that the data are not available.

Methods	Q_C	Q_E	Q_H	Q_3
CNN_BIGRU	76.32	66.43	82.11	76.46
DeepACLSTM	74.70	69.02	78.10	74.81
DCRNN	76.16	66.36	79.60	75.54
DeepCNN	74.46	68.19	80.12	75.21
MUFold-SS	**80.02**	58.63	76.34	74.94
ShuffleNet(CE)	72.20	65.21	83.84	74.81
ShuffleNet_SS(LDAM)	71.62	**70.09**	**83.96**	75.45
Bagging MSA*	–	–	–	71.7
PSSM-Distil*	–	–	–	**76.6**

profile features. Table 10 shows the prediction results for the eight states. As the table shows, the Q_8 prediction accuracy of all predictors on the BC40_MSA_30 dataset is lower than 64%. In particular, the Q_8 prediction accuracy of ShuffleNet_SS (CE) on the validation set is 74.8% while its accuracy on the BC40_MSA_30 test set is only 63.41%. Therefore, low-quality profile features severely reduce the prediction performance of eight-state PSSP.

In order to improve the prediction accuracy of protein chains with low sequence homology, we note that the recently proposed "Bagging MSA" model [53] attempts to enhance low-quality PSSM features based on the "Bagging" mechanism under an unsupervised framework. Moreover, PSSM-Distil [54] further improves Bagging MSA by using a teacher–student network to distill knowledge from high-quality PSSM features with contrastive learning. To compare these two methods, we further give the three-state prediction results on the BC40_MSA_30 dataset, as shown in Table 11. The table shows that the Q_3 prediction accuracy of PSSM-Distil is only 0.14% higher than that of CNN_BIGRU. This means that enhancing low-quality profile features does not provide a significant improvement over the traditional method. It should be noted that the experimental results of Bagging MSA and PSSM-Distil are from the literature [54]; and the training set, validation set, test set and profile features they used are different from our experiment. An objective experimental comparison with Bagging MSA and PSSM-Distil cannot be performed at present since their released source codes cannot be directly used for model training.

5. Conclusion

In this paper, we propose a novel lightweight convolutional network ShuffleNet_SS for sequence-to-sequence PSSP. ShuffleNet_SS is constructed by simply stacking our designed basic network modules. This not only eliminates the impact of padded residue positions on nonpadded residue positions in the forward propagation process of the network by using modified batch normalization but also fully achieves cross-group information exchange through the improved standard channel shuffle operation. The feature reuse ability of ShuffleNet_SS enables it to better capture nonlocal interactions between residues. Moreover, the label distribution aware margin loss is further adopted to enhance the network's ability to learn rare classes. The experimental results on the nine benchmark datasets demonstrate that the proposed ShuffleNet_SS achieves state-of-the-art performance with the fewest parameters compared to the five existing deep predictors. To the best of our knowledge, this is the first time that the loss for imbalanced datasets has been introduced into the field of eight-state deep secondary structure prediction. In the future, we will investigate more imbalanced classification techniques based on deep networks to improve the accuracy of the rare classes 3_{10}-helix (G), isolated β-bridge (B) and π-helix (I) without reducing the overall prediction accuracy.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Science Foundation of China (Grant nos. 61806074, 61771006 and 61976080) and the Science and Technology Foundation of Henan Province of China (no. 212102210387).

References

[1] J. Abbass, J.-C. Nebel, Enhancing fragment-based protein structure prediction by customising fragment cardinality according to local secondary structure, BMC Bioinformatics 21 (1) (2020) 170.
[2] T. Liu, C. Jia, A high-accuracy protein structural class prediction algorithm using predicted secondary structural information, J. Theoret. Biol. 267 (3) (2010) 272–275.
[3] G. Taherzadeh, Y. Zhou, A.W.-C. Liew, Y. Yang, Sequence-based prediction of protein–carbohydrate binding sites using support vector machines, J. Chem. Inf. Model. 56 (10) (2016) 2115–2122.
[4] J.E. Gewehr, R. Zimmer, SSEP-domain: Protein domain prediction by alignment of secondary structure elements and profiles, Bioinformatics 22 (2) (2005) 181–187.
[5] L. Folkman, Y. Yang, Z. Li, B. Stantic, A. Sattar, M. Mort, D.N. Cooper, Y. Liu, Y. Zhou, DDIG-in: Detecting disease-causing genetic variations due to frameshifting indels and nonsense mutations employing sequence and structural properties at nucleotide and protein levels, Bioinformatics 31 (10) (2015) 1599–1606.
[6] R. Aurora, G.D. Rose, Seeking an ancient enzyme in methanococcus jannaschii using orf, a program based on predicted secondary structure comparisons, Proc. Natl. Acad. Sci. 95 (6) (1998) 2818–2823.
[7] J. Pei, B.H. Kim, N.V. Grishin, PROMALS 3D: A tool for multiple protein sequence and structure alignments, Nucleic Acids Res. 36 (7) (2008) 2295–3000.
[8] W. Kabsch, C. Sander, Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features, Biopolymers 22 (12) (1983) 2577–2637.
[9] J. Guo, H. Chen, Z. Sun, Y. Lin, A novel method for protein secondary structure prediction using dual-layer SVM and profiles, Proteins 54 (4) (2004) 738–743.
[10] B. Yang, Q. Wu, Z. Ying, H. Sui, Predicting protein secondary structure using a mixed-modal SVM method in a compound pyramid model, Knowl.-Based Syst. 24 (2) (2011) 304–313.
[11] M.H. Zangooei, S. Jalili, PSSP with dynamic weighted kernel fusion based on SVM-PHGS, Knowl.-Based Syst. 27 (2012) 424–442.
[12] D.T. Jones, Protein secondary structure prediction based on position-specific scoring matrices, J. Mol. Biol. 292 (2) (1999) 195–202.
[13] N. Qian, T.J. Sejnowski, Predicting the secondary structure of globular proteins using neural network models, J. Mol. Biol. 202 (4) (1988) 865–884.
[14] B. Rost, C. Sander, Prediction of protein secondary structure at better than 70, J. Mol. Biol. 232 (2) (1993) 584–599.
[15] K. Asai, S. Hayamizu, K. Handa, Prediction of protein secondary structure by the hidden Markov model, Comput. Appl. Biosci. 9 (1993) 141–146.
[16] Z. Aydin, Y. Altunbasak, M. Borodovsky, Protein secondary structure prediction for a single-sequence using hidden semi-Markov models, BMC Bioinformatics 7 (2006) 178.
[17] J. Martin, J.-F.c. Gibrat, F.c. Rodolphe, Analysis of an optimal hidden Markov model for secondary structure prediction, BMC Struct. Biol. 6 (1) (2006) 25.
[18] S. Salzberg, S. Cost, Predicting protein secondary structure with a nearest-neighbor algorithm, J. Mol. Biol. 227 (2) (1992) 371–374.
[19] W. Yang, K. Wang, W. Zuo, Prediction of protein secondary structure using large margin nearest neighbour classification, Int. J. Bioinform. Res. Appl. 9 (2) (2013) 207–219.
[20] A. Yaseen, Y. Li, Template-based C8-SCORPION: A protein 8-state secondary structure prediction method using structural information and context-based features, BMC Bioinformatics 15 (2014) S3.
[21] 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 (18) (2014) 2592–2597.
[22] J. Zhou, O.G. Troyanskaya, Deep supervised and convolutional generative stochastic network for protein secondary structure prediction, in: The 31st International Conference on Machine Learning, 2014.
[23] M. Spencer, J. Eickholt, J. Cheng, A deep learning network approach to ab initio protein secondary structure prediction, IEEE ACM Trans. Comput. Biol. Bioinform. TCBB 12 (1) (2015) 103–112.
[24] S. Wang, J. Peng, J. Ma, J. Xu, Protein secondary structure prediction using deep convolutional neural fields, Sci. Rep. 6 (2016) 18962.
[25] Z. Li, Y. Yu, Protein secondary structure prediction using cascaded convolutional and recurrent neural networks, in: IJCAI'16 Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, AAAI Press, 2016, pp. 2560–2567.
[26] I. Drori, I. Dwivedi, P. Shrestha, J. Wan, Y. Wang, Y. He, A. Mazza, H. Krogh-Freeman, D. Leggas, K. Sandridge, L. Nan, K.A. Thakoor, C. Joshi, S. Goenka, C. Keasar, I. Pe'er, High quality prediction of protein Q8 secondary structure by diverse neural network architectures, 2018, arXiv preprint arXiv:1811.07143.
[27] Y. Wang, H. Mao, Z. Yi, Protein secondary structure prediction by using deep learning method, Knowl.-Based Syst. 118 (2017) 115–123.
[28] 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 Bioinformatics 20 (1) (2019) 1–12.
[29] 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 Oxford England 33 (18) (2017) 2842–2849.
[30] A. Busia, N. Jaitly, Next-step conditioned deep convolutional neural networks improve protein secondary structure prediction, arXiv abs/1702.03865, 2017.
[31] C. Fang, Y. Shang, D. Xu, MUFOLD-Ss: New deep inception-inside-inception networks for protein secondary structure prediction, Proteins 86 (5) (2018) 592–598.
[32] 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 (2020).
[33] J. Hanson, K. Paliwal, T. Litfin, Y. Yang, Y. Zhou, Improving prediction of protein secondary structure, backbone angles, solvent accessibility and contact numbers by using predicted contact maps and an ensemble of recurrent and residual convolutional neural networks, Bioinformatics 35 (14) (2019) 2403–2410.
[34] M.S. Klausen, M.C. Jespersen, H.B. Nielsen, K.K. Jensen, V.I. Jurtz, C.K. Sonderby, M.O.A. Sommer, O. Winther, M. Nielsen, B. Petersen, NetSurfP-2.0: Improved prediction of protein structural features by integrated deep learning, Proteins 87 (6) (2019) 520–527.
[35] 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.
[36] J. Zhou, H. Wang, Z. Zhao, R. Xu, Q. Lu, CNNHPSS: Protein 8-class secondary structure prediction by convolutional neural network with highway, BMC Bioinformatics 19 (Suppl 4) (2018) 60.
[37] K. Cao, C. Wei, A. Gaidon, N. Arechiga, T. Ma, Learning imbalanced datasets with label-distribution-aware margin loss, in: Advances in Neural Information Processing Systems 32, NeurIPS 2019, 2019.
[38] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the 32nd International Conference on International Conference on Machine Learning, vol. 37, JMLR.org, 2015, pp. 448–456.
[39] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv abs/1704.04861, 2017.
[40] N. Ma, X. Zhang, H.-T. Zheng, J. Sun, ShuffleNet V2: Practical guidelines for efficient CNN architecture design, in: Computer Vision, ECCV 2018, Springer International Publishing, 2018, pp. 122–138.
[41] D. Misra, Mish: A self regularized non-monotonic activation function, 2019, arXiv preprint arXiv:1908.08681.
[42] X. Zhang, X. Zhou, M. Lin, J. Sun, ShuffleNet: An extremely efficient convolutional neural network for mobile devices, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 6848–6856.
[43] Y. Cui, M. Jia, T. Lin, Y. Song, S. Belongie, Class-balanced loss based on effective number of samples, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2019, pp. 9260–9269.
[44] T. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, IEEE Trans. Pattern Anal. Mach. Intell. 42 (2) (2020) 318–327.
[45] J. Wang, W. Zhang, Y. Zang, Y. Cao, J. Pang, T. Gong, K. Chen, Z. Liu, C.C. Loy, D. Lin, Seesaw loss for long-tailed instance segmentation, 2020, arXiv preprint arXiv:2008.10032.
[46] J. Tan, C. Wang, B. Li, Q. Li, W. Ouyang, C. Yin, J. Yan, Equalization loss for long-tailed object recognition, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2020, pp. 11659–11668.
[47] A.K. Menon, S. Jayasumana, A.S. Rawat, H. Jain, A. Veit, S. Kumar, Long-tail learning via logit adjustment, 2020, arXiv preprint arXiv:2007.07314.
[48] G. Wang, R. Dunbrack, PISCES: Recent improvements to a PDB sequence culling server, Nucleic Acids Res. 33 (2005) W94–98, Web Server issue.

[49] J.A. Cuff, G.J. Barton, Application of multiple sequence alignment profiles to improve protein secondary structure prediction, Proteins Struct. Funct. Bioinform. 40 (3) (2000) 502–511.
[50] D.W.A. Buchan, D.T. Jones, The PSIPRED protein analysis workbench: 20 years on, Nucleic Acids Res. 47 (W1) (2019) W402–w407.
[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems, vol. 32, 2019, pp. 8026–8037.
[52] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: International Conference on Learning Representations, 2019.
[53] Y. Guo, J. Wu, H. Ma, S. Wang, J. Huang, Bagging MSA learning: Enhancing low-quality PSSM with deep learning for accurate protein structure property prediction, in: Research in Computational Molecular Biology, Springer International Publishing, 2020, pp. 88–103.
[54] Q. Wang, B. Wang, Z. Xu, J. Wu, P. Zhao, Z. Li, S. Wang, J. Huang, S. Cui, PSSM-Distil: Protein secondary structure prediction (PSSP) on low-quality PSSM by knowledge distillation with contrastive learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 617–625.