scientific reports

OPEN

Parameterized hypercomplex convolutional network for accurate protein backbone torsion angle prediction

Wei Yang, Shujia Wei & Lei Zhang✉

Predicting the backbone torsion angles corresponding to each residue of a protein from its amino acid sequence alone is a challenging problem in computational biology. Existing torsion angle predictors mainly use profile features, which are generated by performing time-consuming multiple sequence alignments, for torsion angle prediction. Compared with traditional profile features, embedding features from pretrained protein language models have significant advantages in prediction performance and computational speed. However, embedding features usually have higher dimensions and different embedding features have significantly different dimensions. To this end, we design a novel parameter-efficient deep torsion angle predictor, PHAngle, specifically for embedding features. PHAngle is a parameterized hypercomplex convolutional network consisting of parameterized hypercomplex linear and convolutional layers whose weight parameters can be characterized as the sum of Kronecker products. Experimental results on six benchmark test sets including TEST2016, TEST2018, TEST2020_HQ, CASP12, CASP13 and CASP-FM demonstrate that PHAngle achieves the state-of-the-art torsion angle performance with the fewest parameters compared to the nine existing methods. The source code and datasets are available at https://github.com/fengtuan/PHAngle.

Keywords Torsion angle prediction, Parameterized hypercomplex convolutional network, Pretrained protein language model, Embedding feature

The protein backbone torsion angles (ϕ and ψ) are an important protein structural property. For each protein, the bond lengths and bond angles between adjacent backbone atoms are fixed, while the variable torsion angles define its backbone structure. The torsion angles determine the overall shape and stability of proteins and affect various interactions within a protein molecule. Moreover, the torsion angles have a high correlation with the secondary structures of proteins (especially helical and sheet structures), and this correlation helps to determine the local conformation of proteins. In particular, the torsion angles can also help define highly variable loop conformations. Accurately predicted torsion angles can not only greatly reduce conformational search space and thereby aid in reconstructing the three-dimensional structure of proteins[1], but can also be applied to improve protein fold recognition[2] and sequence alignment[3].

Since protein torsion angles are continuous real values from -180 to 180 degrees, torsion angle prediction is usually regarded as a regression problem. For a given protein, the goal of torsion angle prediction is to predict the two torsion angles ϕ and ψ corresponding to each residue in it based only on its amino acid sequence. According to the data form of the prediction output, existing torsion angle prediction algorithms can be divided into two categories: discrete-valued torsion angle prediction and continuous-valued torsion angle prediction. For discrete-valued torsion angle prediction, continuous torsion angle values are converted into discrete class states to reduce the difficulty of prediction. Representative methods include SHIFTOR[4], DHPRED[5] and SpiderBin[6]. For continuous-valued torsion angle prediction, it can be further categorized into sliding-window based prediction and sequence-to-sequence prediction. The sliding window-based prediction algorithms assume that the two torsion angles corresponding to central amino acid residue are determined by all the amino acids in the local window in which they are located. In particular, the performance of these algorithms depends on the size of the sliding window. Smaller windows usually cannot obtain better prediction results because they do not contain enough information, while larger windows increase the computational cost of the prediction model.

Henan Key Laboratory of Big Data Analysis and Processing, Henan Engineering Laboratory of Spatial Information Processing, School of Computer and Information Engineering, Henan University, Kaifeng 475004, China. ✉email: zhangleicshnu@163.com

For the sliding window-based algorithms, the pioneering work is Real-SPINE[7] proposed by Dor et al., which combines the outputs of two multilayer perceptron predictors for torsion angle prediction. Subsequently, Wu et al. proposed ANGLOR[8] by combining support vector machine and multilayer perceptron. SPINE X[9] used a multistep learning strategy for torsion angle prediction based on multilayer perceptron. TANGLE[10] adopted a two-level support vector regression method to perform real-value torsion angle prediction. In addition, other method variants have been proposed to perform sliding window-based torsion angle prediction[11–15].

Unlike the sliding window-based prediction, sequence-to-sequence prediction algorithms perform torsion angle prediction based on the entire amino acid sequence of a given protein chain. This approach has two advantages: (1) it can capture both local and global interactions between residues during prediction since the full information of the amino acid sequence can be exploited; (2) it eliminates the need to perform time-consuming sliding-window operations in predicting the torsion angles. Currently, deep learning techniques are mainly used to perform sequence-to-sequence torsion angle prediction. DeepRIN[16] employed deep inception residual convolutional network and hybrid profile features for protein torsion angle prediction. Based on a multitask learning strategy, SPOT-1D[17] designed an ensemble model of bidirectional recurrent and residual convolutional neural networks to simultaneously predict backbone torsion angles, secondary structure, half-sphere exposure, contact numbers and solvent accessible surface area. OPUS-TASS[18] proposed a deep prediction model that integrates a convolutional network module, a bidirectional LSTM module and a Transformer module. ESIDEN[19] developed four novel features from evolutionary signatures and the Ramachandran basins to improve the prediction performance of torsion angles. It should be noted that the main feature adopted by the above methods is the position-specific scoring matrix (PSSM). However, generating PSSM profile features requires performing time-consuming multiple sequence alignments against a large-scale protein sequence database. Recently, pretrained protein language models such as ProtTrans[20] and ESM-1b[21] have been shown to be able to learn useful biological information from large-scale protein sequence data based on self-supervised learning techniques. In particular, embedding features derived from these pretrained models have achieved better performance in several downstream tasks. As a result, some works including SPOT-1D-LM[22], NetSurfP-3.0[23] and SAINT-Angle[24] have started to employ embedding features for torsion angle prediction. For a more detailed review of protein torsion angle prediction can be found in the literature[25].

Recently, hypercomplex neural networks have gained much attention in the field of deep learning due to their better theoretical properties and their ability to maintain valuable network performance with significantly fewer parameters. To generalize the hypercomplex multiplication to arbitrary nD hypercomplex space, Zhang et al.[26] designed a novel parameterized hypercomplex linear layer based on the sum of Kronecker products. Subsequently, Grassucci et al.[27] proposed a parameterized hypercomplex convolutional layer and devised some lightweight and more efficient large-scale hypercomplex models. Nowadays, hypercomplex neural networks have been successfully applied in many domains such as Image-to-Image translation[28], graph classification[29], speech recognition[30] and breast cancer classification[31].

In this study, we devise a novel parameterized hypercomplex convolutional network architecture, called PHAngle, for protein torsion angle prediction. PHAngle is composed of parameterized hypercomplex linear and convolutional layers whose weight parameters can be characterized as the sum of Kronecker products. In particular, to ensure the consistency of the torsion angle prediction results under different batch sizes, we introduce 1-dimensional batch normalization with a mask matrix, which eliminates the impact of padded residue positions on nonpadded residue positions during network forward propagation. Experimental results on six benchmark test sets demonstrate that PHAngle achieves the state-of-the-art torsion angle performance with the fewest parameters compared to the nine existing methods. In addition, we compare the performance difference between direct and indirect torsion angle prediction by using three different loss functions.

Methods
Overview
In this section, we design a novel parameter-efficient convolutional network model to perform protein torsion angle prediction. In particular, by using parameterized hypercomplex linear and convolutional layers instead of the traditional fully-connected and convolutional layers, respectively, in the designed network model, we significantly reduce the parameters of the model while maintaining the prediction performance. We named the proposed torsion angle prediction method PHAngle. In the following, we will first introduce two parameterized hypercomplex network layers based on Kronceker product, then describe the proposed protein torsion angle prediction framework, and finally give the loss function used for model training.

Parameterized hypercomplex linear and convolutional layers
In this subsection, we will introduce two parameterized hypercomplex network layers: a hypercomplex linear layer and a hypercomplex convolutional layer. In particular, they are able to better capture the internal latent relationships of multidimensional data compared to traditional fully connected and convolutional layers.

Parameterized hypercomplex liner layer The parameterized hypercomplex linear layer[26] is defined as:

$$\mathbf{PHLinear}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}, \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^d$ is an input vector. Note that different from the traditional fully connected layer, the weight matrix $\mathbf{W} \in \mathbb{R}^{m\times d}$ is obtained by computing the sum of R Kronceker products:

$$\mathbf{W} = \sum_{r=1}^{R} \mathbf{A}_r \otimes \mathbf{B}_r, \tag{2}$$

where the symbol $\otimes$ denotes the Kronecker product, *d* is the dimension of input features, *m* is the dimension of output features, the positive integer *R* controls the number of parameters used to represent the weight matrix, and the matrixes $\mathbf{A}_r \in \mathbb{R}^{(m/n)\times n}$ and $\mathbf{B}_r \in \mathbb{R}^{n\times(d/n)}$ and the bias vector $\mathbf{b} \in \mathbb{R}^m$ are learnable parameters. In particular, both *m* and *d* are assumed to be divisible by the hyperparameter $n \in \mathbb{Z}_{>0}$. Note that this Kronecker product-based weight representation allows PHLinear to capture intra- and inter-channel correlations.

Parameterized hypercomplex convolutional layer Here we only introduce the 1-dimensional hypercomplex convolutional layer[27], which can be formalized as:

$$\mathbf{PHConv1D}(\mathbf{x}) = \mathbf{K} * \mathbf{x} + \mathbf{b}, \tag{3}$$

where the kernel tensor $\mathbf{K} \in \mathbb{R}^{d\times m\times s}$ is computed as follows:

$$\mathbf{K} = \sum_{r=1}^{R} \mathbf{C}_r \otimes \mathbf{D}_r, \tag{4}$$

where *d* is the dimension of the input channel, *m* is the dimension of the output channel, and *s* is the convolutional kernel size. The tensors $\mathbf{C}_r \in \mathbb{R}^{(d/n)\times n\times s}$ and $\mathbf{D}_r \in \mathbb{R}^{n\times(m/n\,)\times s}$ and the bias vector $\mathbf{b} \in \mathbb{R}^m$ are parameters that need to be learned during training.

Initialization of parameters Appropriate parameter initialization methods are very important for the convergence of deep models. In this study, the parameter *n* is set as $2^{\lfloor \log_2 \sqrt{\max(d,m)} \rfloor}$ and the hyperparameter *R* is determined by the performance on the validation set. For PHLinear and PHConv1D, we need to initialize the two parameter sets $\{A_r, B_r\}_{r=1}^R$ and $\{C_r, D_r\}_{r=1}^R$, which implicitly determine the initial values of **W** and **K**. Obviously, such implicitly determined initial values should be close to those obtained by the initialization strategies such as kaiming_uniform that are usually adopted for standard fully connected and convolutional layers, since the latter have been proven to be effective in practice and theory. To this end, we first generate **W** and **K** using kaiming_uniform or other random initialization methods, and then determine the parameter sets $\{A_r, B_r\}_{r=1}^R$ and $\{C_r, D_r\}_{r=1}^R$. Without loss of generality, let $\mathbf{W}_0 \in \mathbb{R}^{m\times d}$ be a matrix randomly generated based on a uniform distribution $U\left(-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}}\right)$. To obtain the initial values of parameter set $\{A_r, B_r\}_{r=1}^R$, we solve the following multidimensional nearest Kronecker product problem[32]:

$$\min_{\{\mathbf{A}_r,\mathbf{B}_r\}_{r=1}^R} \left\| \mathbf{W}_0 - \sum_{r=1}^{R} \mathbf{A}_r \otimes \mathbf{B}_r \right\|_F^2, \tag{5}$$

where $\|\cdot\|_F^2$ represents the square of the Frobenius norm of a matrix. By converting $\mathbf{W}_0$ to $\tilde{\mathbf{W}}_0 \in \mathbb{R}^{m\times d}$ using the rearrangement operation in the literature[33], the optimization problem in Eq. (5) can be equivalently transformed into the following form:

$$\arg\min_{\{\mathbf{A}_r,\mathbf{B}_r\}_{r=1}^R} \left\| \tilde{\mathbf{W}}_0 - \sum_{r=1}^{R} vec(\mathbf{A}_r) vec(\mathbf{B}_r)^T \right\|_F^2, \tag{6}$$

where *vec*(.) denotes stacking the elements of a matrix into a vector in row-major order. In particular, the optimization problem in Eq. (6) can be solved using singular value decomposition. Specifically, let the singular value decomposition of matrix $\tilde{\mathbf{W}}_0$ be $\tilde{\mathbf{W}}_0 = \sum_{r=1}^{\min\{m,d\}} \sigma_r u_r v_r$, where σ_r is a singular value, and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min\{m,d\}}$, and u_r and v_r are called left-singular vector and right-singular vector of σ_r, respectively. Then the initial values of matrices $\mathbf{A}_r$ and $\mathbf{B}_r$ can be computed by:

$$\begin{cases} \mathbf{A}_r = \phi_{(m/n)\times n}\left(\sqrt{\sigma_r} u_r\right) \\ \mathbf{B}_r = \phi_{n\times(d/n)}\left(\sqrt{\sigma_r} v_r\right) \end{cases}, \tag{7}$$

where $\phi\,(\cdot)$ is the inverse operation of *vec*$(\cdot)$ that restores a vector back into a matrix of specified dimensions. For the parameter set $\{C_r, D_r\}_{r=1}^R$, their initial values can be determined by the same method based on *s* randomly generated matrices.

The protein torsion angle prediction framework

The proposed protein torsion angle prediction framework is shown in Fig. 1. The input of our network framework includes two parts: a feature tensor $X \in \Re^{N\times L\times C_{in}}$ and a binary mask matrix $M \in \Re^{N\times L}$, where *N* denotes the number of protein chains in a minibatch, *L* denotes the maximum length of *N* protein chains, and C_{in} denotes the dimension of the input features. For the binary mask matrix, the entry $M_{ij} = 1$ if the *j*th residue in the *i*th protein chain is a nonpadded residue position, and 0 otherwise. The purpose of introducing the binary matrix **M** is to eliminate the impact of padded residue positions on nonpadded residue positions during the forward propagation process. As shown in Fig. 1, the binary matrix **M** is fed into both the backbone network and the "Masked BN" layer. Here, "Masked BN" denotes 1D batch normalization with a mask matrix[34], which

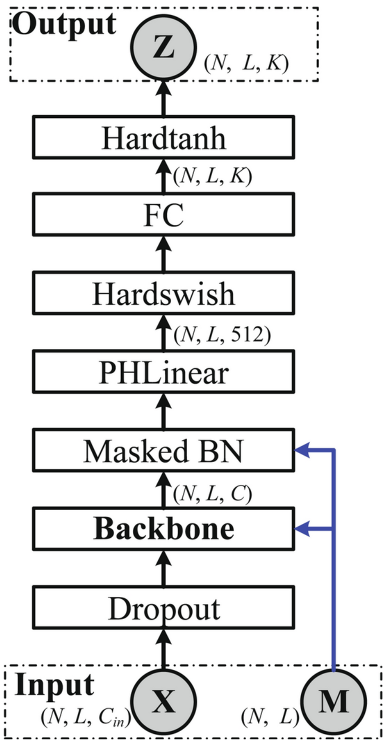

Fig. 1. The framework for protein torsion angle prediction.

first uses the features corresponding to the nonpadded positions to calculate the mean and variance, and then further converts the feature vectors corresponding to all padded positions into zero vector after performing affine transformation. Note that the output channel size of the backbone network is *C*. Moreover, the output of the "Masked BN" layer is passed through a PHLinear layer and a fully connected layer (FC). The PHLinear layer is followed by a Hardswish activation layer. The FC uses *K* output units to predict the torsion angles ϕ and ψ, and its output is further fed into the function Hardtanh to limit the predicted values of torsion angles between -1 and 1. In particular, *K* = 4 or 2, and its specific value depends on the loss function employed. If the target values of the loss function are the sines and cosines of the two torsion angles, then *K* = 4. Otherwise, *K* = 2, which means that the network directly predicts the values of the two torsion angles. In addition, the backbone network is built based on the parameterized hypercomplex linear and convolutional layers. We will describe it in the following subsection.

The backbone network
To build the convolution-based backbone, we first designed a building block, PHBlock, as shown in Fig. 2. The input of PHBlock includes two parts: a feature tensor $X \in \Re^{N\times M_{in}\times L}$ and a binary mask matrix **M**, where M_{in} represents the input channel size. The feature tensor **X** is fed into the first convolutional block with residual connection, which consists of a "masked BN" layer, a PHConv1D layer with **M** filters of kernel size 1, and a Hardswish activation layer. In particular, when $M_{in} \neq M$, the dashed line in Fig. 2 represents a linear transformation that converts the channel size from M_{in} to *M*. Otherwise, it represents an identity mapping. In this study, we use PHConv1D with kernel size 1 to perform the linear transformation. The output of the first convolutional block is then divided equally into two parts along the channel dimension. The first part is kept unchanged and the second part is further fed into a dropout layer, which is followed by the second convolutional block with residual connection. In the second convolutional block, the kernel size and the output channel size are set to 3 and *M*/2, respectively. Finally, the outputs of the two branches are merged using a concatenate operation and the merged result is further fed into another dropout layer to overcome the overfitting.

Building on the designed block PHBlock, we present the architecture of backbone network in Fig. 3. The backbone employs *l* stacked dual-branch modules to capture nonlocal and local interactions between residues. In each dual-branch module, one branch contains only one PHBlock and the other branch contains two stacked PHBlocks with residual connection. Moreover, a concatenate operation is used to merge the outputs of the two branches. Note that the output channel size of all dual-branch modules is *C*, which implies that $M = C/2$ and *C* must be divisible by 4. The hyperparameters *C* and *l* control the capacity of the backbone.

Loss function
The torsion angles ϕ and ψ are the dihedral angles formed by four consecutive atoms $C^{(i-1)} - N^{(i)} - C_\alpha^{(i)} - C^{(i)}$ and $N^{(i)} - C_\alpha^{(i)} - C^{(i)} - N^{(i+1)}$ in the protein backbone, respectively. The value range of torsion angles is $[-180^\circ$ $,180^\circ]$. Note that the proposed method can perform direct torsion angle prediction by outputting the value of

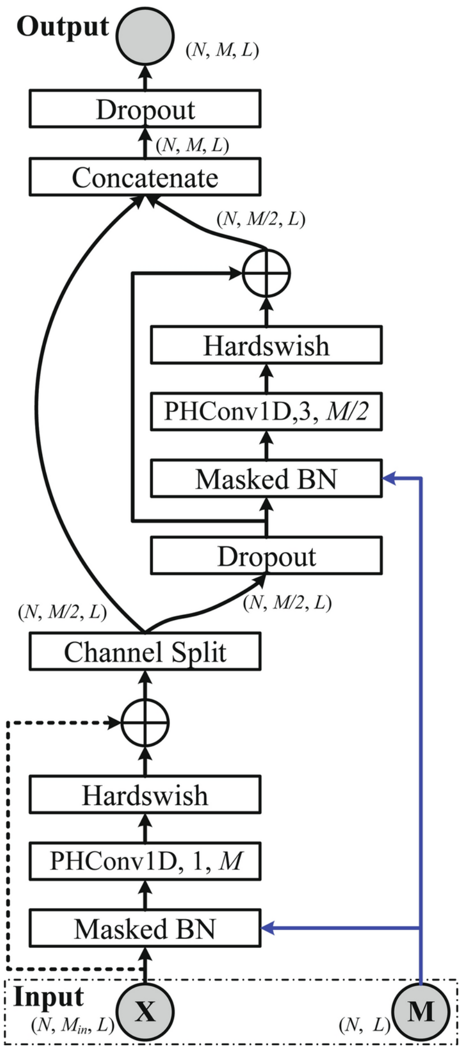

Fig. 2. The building block PHBlock.

the torsion angle or indirect torsion angle prediction by outputting both the sine and cosine values of the torsion angle. In both cases, the range of network output values is [-1,1].

For direct prediction, let *y* and *z* be the target and predicted values of a given torsion angle, respectively. Note that *y* belongs to -180 to 180, while *z* belongs to -1–1. To eliminate the problem of periodicity of angles, the loss function for a single angle can be defined as:

$$\rho\left(y,z\right)=\min\left(\left|z-\frac{y}{180}\right|,2-\left|z-\frac{y}{180}\right|\right) \tag{8}$$

Due to the characteristics of structure determination methods, the coordinates of some atoms in proteins with known structures may be missing. For example, flexible regions are not observable in X-ray crystallization experiments. This will result in missing actual values for some torsion angles in proteins with known structures. In addition, the torsion angles ϕ and ψ are undefined at the beginning and end of a protein chain, respectively. Obviously, these torsion angles should be ignored during model training and evaluation. To this end, we set the target value of these torsion angles to NaN to facilitate filtering. Furthermore, to construct a minibatch containing *N* protein chains, we need to pad several residues on the right of the shorter protein chains to make their lengths reach the maximum length of the *N* protein chains. In particular, we set the feature vector and target value corresponding to a padding position to zero vector and NaN, respectively. For a given minibatch, let $Y\in\Re^{N\times L\times 2}$ be its corresponding target tensor and $Z\in\Re^{N\times L\times 2}$ be the corresponding predicted output. Then the loss function for direct prediction can be defined as:

$$\xi_{\text{direct}}=\sum_{k=0}^{1}\left(\frac{\sum_{i=0}^{N-1}\sum_{j=0}^{L-1}\mathbb{1}\left\{Y_{ijk}\neq\text{NaN}\right\}\rho\left(Y_{ijk},Z_{ijk}\right)}{2\sum_{i=0}^{N-1}\sum_{j=0}^{L-1}\mathbb{1}\left\{Y_{ijk}\neq\text{NaN}\right\}}\right), \tag{9}$$

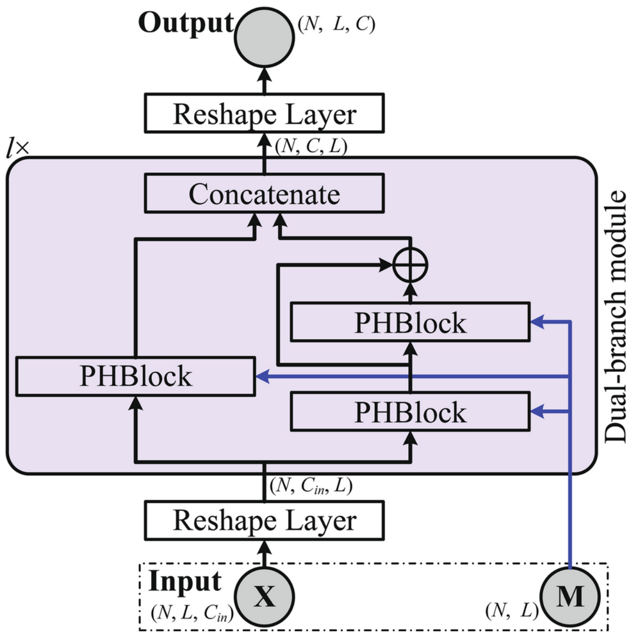

Fig. 3. The architecture of backbone network.

Dataset	#Chains	#Residues	#Max	#Min
TEST2016	1213	287877	699	30
TEST2018	250	56654	615	31
TEST2020_HQ	121	18978	601	30
CASP12	55	10283	462	55
CASP13	32	5354	718	31
CASPFM	56	8100	356	67

Table 1. Statistics for the six test sets. #Chains = Number of protein chains; #Residues =Number of residues; #Max = Maximum chain length; #Min = Minimum chain length.

where the variable *k* is used to index the two torsion angles $\phi(k=0)$ and $\psi(k=1)$, and the indicator function $\mathbb{1}\{Y_{ijk} \neq \mathrm{NaN}\}$ outputs 1 if $Y_{ijk} \neq \mathrm{NaN}$; otherwise, it outputs 0.

For indirect prediction, the prediction network needs to provide two output values for each torsion angle. Assuming that z_1 and z_2 correspond to the sine and cosine output values of a given torsion angle, respectively, then its predicted value can be computed by $\tan^{-1}(z_1/z_2)$. For a given minibatch, let $\bar{Z} \in \Re^{N\times L\times 4}$ be the corresponding predicted output. Then the loss function for indirect prediction can be defined as:

$$\xi_{\text{indirect}} = \sum_{k=0}^{1}\left(\frac{\sum_{i=0}^{N-1}\sum_{j=0}^{L-1}\mathbb{1}\{Y_{ijk}\neq \mathrm{NaN}\}\left[\left|\sin(Y_{ijk})-\bar{Z}_{ij(2k)}\right|^{\alpha}+\left|\cos(Y_{ijk})-\bar{Z}_{ij(2k+1)}\right|^{\alpha}\right]}{2\sum_{i=0}^{N-1}\sum_{j=0}^{L-1}\mathbb{1}\{Y_{ijk}\neq \mathrm{NaN}\}}\right), \tag{10}$$

where we limit α to 1 or 2, with $\alpha = 1$ indicating that the mean absolute error is used, and $\alpha = 2$ indicating that the mean squared error is used.

Experimental results

Datasets

In this study, we adopt the widely used nonhomologous dataset SPOT-1D[17] to train the proposed method. The dataset SPOT-1D consists of a training set containing 10029 protein chains and a validation set containing 983 protein chains, and all protein chains do not exceed 700 amino acid residues in length. In particular, these protein chains were culled from the PISCES server on February 2017 with the constraints of resolution < 2.5 Å, R-free < 1, and percent sequence identity $\leq$ 25% according to BlastClust. Moreover, to evaluate the proposed method, we conduct experiments on six publicly available test sets: TEST2016, TEST2018, TEST2020_HQ, CASP12, CASP13 and CASP-FM. Table 1 summarizes the detailed dataset statistics. TEST2016 consists of 1213 protein chains released between June 2015 and February 2017 in the PDB database. TEST2018 is composed of 250 high-quality protein chains with a resolution of < 2.5 Å and R-free < 0.25 that were released between January 2018 and July 2018 in the PDB database. Both TEST2016 and TEST2018 have $\leq$ 25% sequence identity with the training and validation sets. TEST2020_HQ[22] consists of 121 protein chains released between May 2018 and April 2020, which was obtained by first removing close and remote homologs of all proteins released before 2018 using HMM models, followed by filtering using the same constraints as for TEST2018, and finally deleting protein chains greater than 700 in length. The remaining three test sets CASP12, CASP13, and CASP-FM were collected from the website http://predictioncenter.org/ by the literature[35], where CASP stands for Critical Assessment of protein Structure Prediction and FM stands for template-Free Modeling. Note that homology reduction has been performed to remove sequences that have more than 25% sequence similarity to the sequences in the training set. There are 55 protein chains in CASP12, 32 protein chains in CASP13 and 56 protein chains in CASP-FM. For the 56 FM proteins in CASP-FM, 8 of them are from CASP10, 16 from CASP11, 22 from CASP12, and 10 from CASP13.

Feature representation

To predict protein torsion angles based on amino acid sequences, each amino acid residue in the sequence needs to be converted into a numerical vector. In this study, we adopt eight feature representations to perform such conversion: one base feature representation and seven embedding feature representations. For the base feature representation, the feature matrix of each protein chain consists of PSSM profile feature, HMM profile feature and physicochemical property feature. In particular, for a protein chain of length L, its PSSM profile feature is a matrix of size $L \times 20$, which was generated by performing three iterations of the PSI-BLAST program with default parameters against the UniRef90 database updated in April 2018. The corresponding HMM profile feature is a matrix of size $L \times 30$ generated by running HHblitsv3.0.3 against the Uniprot database updated in October 2017. The physicochemical property feature is a matrix of size $L \times 7$ involving seven property values for each amino acid: hydrophobicity, van der Waals volume, isoelectric point, sheet probability, helix probability, polarizability and graph shape index. Thus, the base feature is a matrix of size $L \times 57$. Moreover, for each embedding feature representation, its feature matrix is generated by feeding the amino acid sequence of a given protein chain into a pretrained protein language model, which is obtained by training a Transformer model on a large-scale non-redundant protein sequence dataset using self-supervised learning techniques. To investigate the impact of different embedding feature representations on torsion angle prediction performance, we constructed seven different embedding features based on seven pretrained protein language models prot_t5_xl_uniref50[20], ankh-base[36], ankh-large[36], esm-1b[21], esm2_t33_650M_UR50D[37], esm2_t36_3B_UR50D[37] and esm2_t48_15B_UR50D[37]. In particular, the embedding feature dimensions of these models are 1024, 768, 1536, 1280, 1280, 2560, and 5120, respectively. It should be noted that, among the above seven pretrained protein language models, the first three use the T5 model based on the encoder-decoder architecture, while the last four employ the BERT model based on the encoder architecture. For the T5 models, we only use its encoder to generate embedding features.

Experimental settings

All our experiments are conducted based on PyTorch. For training, we use the AdamW optimizer with learning rate 1e-3 and weight decay 1e-5. The batch size is set to 32. In particular, the indirect prediction loss with $\alpha = 2$ and the embedding representation generated by ankh-large will serve as our default loss function and feature, respectively, if not specified. For the proposed method, the default values of its hyperparameters R and l are set to 96 and 2, respectively. The dropout ratio is set to 0.2. For embedding features with feature dimension greater than 1024, the output channel parameter C is set to its feature dimension, otherwise C is set to 1024. To evaluate the performance of different protein torsion angle prediction methods, we adopt the widely used evaluation metric mean absolute error (MAE). Let y and z be the observed and predicted values of a given torsion angle, respectively, then its absolute error is defined as $\min\left(|z-y|, 360-|z-y|\right)$. Note that at this point both y and z are in the range [-180, 180]. For a given test set, the final reported MAE is the average of the absolute errors across all valid torsion angles. Moreover, to alleviate overfitting and reduce training costs, we adopt an early stopping strategy with a patience value of 5 based on the average MAE of two torsion angles on the validation set.

Comparison to state-of-the-art methods

In this section, we compare PHAngle with the nine representative torsion angle prediction networks BiLSTM[22], MS-ResNet[22], MS-Res-LSTM[22], OPUS-TASS[18], ESIDEN[19], NetSurfP3.0[23], DeepRIN[16], SAINT-Angle (ProtTrans)[24] and SAINT-Angle (Residual)[24] on the six test sets, TEST2016, TEST2018, TEST2020_HQ, CASP12, CASP13, and CASP-FM. Among the nine prediction networks, the first three networks were first proposed in the literature[38] and were subsequently used to construct the ensemble prediction methods SPOT-1D[17] and SPOT-1D-LM[22]. BiLSTM is a two-layer bidirectional Long Short-Term Memory network (LSTM). MS-ResNet is a multiscale residual network with a pre-activation architecture. MS-Res-LSTM is a hybrid network consisting of bidirectional LSTM and multiscale residual network. OPUS-TASS is a hybrid network stacked by a convolutional network, a Transformer network and a bidirectional LSTM. ESIDEN is a prediction network consisting of three LSTM modules. NetSurfP3.0 is constructed by stacking two 1-dimensional convolutional layers with huge convolutional kernels (129 and 257) and two layers of bidirectional LSTM. DeepRIN is a deep residual inception network architecture capable of capturing local and global interactions between amino acids. SAINT-Angle (ProtTrans) is a network based on the attention augmented inception-inside-inception (2A3I) module[35], while SAINT-Angle (Residual) adopts a novel RES-2A3I module, which is obtained by extending the 2A3I module by placing residual connections in each of the inception and self-attention modules. For a fair comparison, all methods should use the same training data and feature representation. Therefore, we trained all existing networks according to their default training settings and the SPOT-1D dataset, and used the embedding

	TEST2016		TEST2018		TEST2020_HQ	
Method	**MAE(ϕ)**	**MAE(ψ)**	**MAE(ϕ)**	**MAE(ψ)**	**MAE(ϕ)**	**MAE(ψ)**
BiLSTM	15.37	21.37	15.94	22.80	18.24	29.06
MS-ResNet	15.22	21.09	15.90	22.75	18.45	29.41
MS-Res-LSTM	15.17	21.06	15.77	22.59	18.21	29.07
OPUS-TASS	14.97	20.78	15.54	22.29	18.15	29.00
ESIDEN	15.66	21.20	16.17	22.64	18.62	28.98
NetSurfP3.0	15.01	20.64	15.67	22.12	18.24	28.96
DeepRIN	14.96	20.63	15.62	22.21	18.20	29.49
SAINT-Angle(ProtTrans)	15.28	21.13	15.87	22.73	18.36	29.26
SAINT-Angle(Residual)	15.15	20.84	15.77	22.44	18.27	28.93
PHAngle	14.60	**20.16**	**15.30**	**21.78**	**17.91**	**28.58**
PHAngle(standard)	**14.54**	20.19	15.38	21.91	18.06	29.19

Table 2. Comparison of prediction results on the three larger test sets TEST2016, TEST2018, and TEST2020_HQ. Best results are shown in boldface.

	CASP12		CASP13		CASP-FM	
Method	**MAE(ϕ)**	**MAE(ψ)**	**MAE(ϕ)**	**MAE(ψ)**	**MAE(ϕ)**	**MAE(ψ)**
BiLSTM	18.18	26.23	17.5	24.83	19.95	30.89
MS-ResNet	18.09	26.11	17.55	24.43	20.02	31.27
MS-Res-LSTM	17.98	25.76	17.46	24.32	19.84	30.19
OPUS-TASS	17.96	25.45	17.42	23.85	19.87	**30.17**
ESIDEN	18.35	26.04	17.31	24.33	20.12	30.79
NetSurfP3.0	18.03	25.46	17.48	23.96	19.86	30.87
DeepRIN	17.87	25.23	17.2	24.35	19.88	30.51
SAINT-Angle(ProtTrans)	18.26	26.45	17.38	23.99	19.9	30.89
SAINT-Angle(Residual)	17.94	26.1	17.41	24.09	19.95	30.82
PHAngle	17.78	**25.22**	**17.13**	**23.55**	**19.64**	30.19
PHAngle(standard)	**17.70**	25.38	17.31	23.71	19.81	30.50

Table 3. Comparison of prediction results on the three smaller test sets CASP12, CASP13, and CASP-FM. Best results are shown in boldface.

features generated by the pretrained protein language model ankh-large as feature inputs for each network. In particular, to study the impact of network architecture on torsion angle prediction performance, we did not use techniques such as ensemble learning and multitask learning.

The comparison results on the six test sets are given in Tables 2 and 3. As can be seen from Table 2, the performance of the proposed method PHAngle on the three larger test sets TEST2016, TEST2018 and TEST2020_HQ is significantly better than that of the nine existing methods. On the largest test set TEST2016, PHAngle's mean absolute errors for torsion angles ϕ and ψ are 0.36 and 0.47 lower than those of the second best method DeepRIN, respectively. On the three smaller test sets CASP12, CASP13, and CASP-FM, PHAngle outperforms other nine existing methods in most cases. Note that although SAINT-Angle (Residual) is a complex convolutional network incorporating inception-inside-inception module, attention mechanism, and residual connection, it is obviously inferior to the relatively simple networks DeepRIN and PHAngle in terms of torsion angle prediction performance. This shows that blindly increasing the complexity of the network does not improve the prediction accuracy of the torsion angle. In addition, Table 4 lists the model sizes of the proposed method with the nine state-of-the-art methods. As can be seen from the table, the model size of PHAngle is only 14.6 MB, which is significantly smaller than that of the other nine methods. This means that the proposed method achieves state-of-the-art performance with fewest parameters.

In addition, for the proposed method PHAngle, we can replace network layers PHLinear and PHConv1D in its backbone network using standard fully connected layers and 1-dimensional convolutional layers, respectively. In particular, we refer to the replaced method as PHAngle(standard). As can be seen from Tables 2 and 3, PHAngle (standard) is only slightly better than PHLinear in predicting the torsion angle ϕ of the test sets TEST2016 and CASP12, while the latter outperforms the former in most cases. Moreover, it should be noted that the model size of PHAngle (standard) is 56.5 MB, which is 3.87 times the size of the PHAngle model. This implies that the proposed parameter-efficient model maintains and even improves the torsion angle prediction performance while reducing the network parameter redundancy.

In order to objectively evaluate the proposed method, we further executed a 10-fold cross-validation experiment. Specifically, we first merged the three datasets (the training and validation sets of SPOT-1D and

Methods	Model size (MB)
BiLSTM	196.9
MS-ResNet	105.3
MS-Res-LSTM	87.6
OPUS-TASS	356.7
ESIDEN	44.7
NetSurfP3.0	262.7
DeepRIN	22.7
SAINT-Angle(ProtTrans)	59.3
SAINT-Angle(Residual)	51.8
PHAngle	14.6
PHAngle(standard)	56.5

Table 4. Comparison of the model size of the proposed method with state-of-the-art methods.

Methods	MAE(ϕ)	MAE(ψ)
BiLSTM	14.63±0.10	19.71±0.21
MS-ResNet	14.89±0.10	20.06±0.23
MS-Res-LSTM	14.89±0.09	20.08±0.23
OPUS-TASS	14.66±0.08	19.77±0.22
ESIDEN	15.13±0.14	20.09±0.23
NetSurfP3.0	14.69±0.09	19.79±0.21
DeepRIN	14.72±0.10	19.86±0.20
SAINT-Angle(ProtTrans)	14.98±0.10	20.09±0.22
SAINT-Angle(Residual)	14.79±0.11	19.87±0.21
PHAngle	**14.40±0.09**	**19.40±0.20**

Table 5. Experimental results of the 10-fold cross-validation. Best results are shown in boldface.

TEST2016) into a single dataset, which contains 12225 protein chains. Then, we randomly divided this dataset into 10 subsets, where each of the first 5 subsets contained 1223 protein chains and each of the last 5 subsets contained 1222 protein chains. Finally, we iteratively select each subset as the test set. In particular, for the 9 subsets left after selecting a test set, we use one of them as the validation set and put the other 8 together as the training set. Table 5 gives the mean and standard deviation of the proposed method and nine state-of-the-art methods on the 10-fold cross-validation experiment. As can be seen from the table, PHAngle consistently outperforms the other nine methods.

Ablation study

To analyze the impact of feature representation, hyperparameter R, channel size, network depth and loss function on the performance of the proposed method, we conduct extensive experiments on the TEST2016 test set and validation set. In particular, we only change the parameters of interest in each experiment, while the other hyperparameters keep their default values.

Impact of feature representation To investigate the impact of different feature representations on the torsion angle prediction performance of the proposed method, we perform comparative experiment on the test set TEST2016. Note that the dimension of the base feature is 57, which is significantly lower than that of the seven embedding features. For the sake of fairness in comparison, we first project it to 1024 dimensions using a fully connected layer, and then feed the projected feature into our backbone network. In particular, we use the name of the pretrained models to indicate the type of embedding features. The mean absolute errors of the proposed method under eight feature representations for torsion angles ϕ and ψ are shown in Fig. 4. As can be seen from the figure, the embedding feature esm-1b obtains the worst prediction performance, and its prediction errors on both torsion angles are consistently higher than those of other seven feature representations. In terms of prediction error, the base feature is only marginally better than esm-1b, but significantly inferior to the other six embedding features. Among the eight feature representations, ankh-large achieves the best prediction performance, which is why we use it as the default feature representation. The mean absolute errors of anke-large on the torsion angles ϕ and ψ are 0.51 and 0.8 lower than those of esm2_t48_15B_UR50D, respectively. Note that the size of the pretrained model esm2_t48_15B_UR50D is 60.5 GB, while the size of ankh-large is 11.4 GB. This suggests that the embedding features generated by the larger pretrained model are not guaranteed to yield better torsion angle prediction performance. Moreover, the prediction performance of the other five embedding features esm2_t36_3B_UR50D, esm2_t33_650M_UR50D, ankh-base, prot_t5_xl_uniref50, and esm-1b decreases in this order.

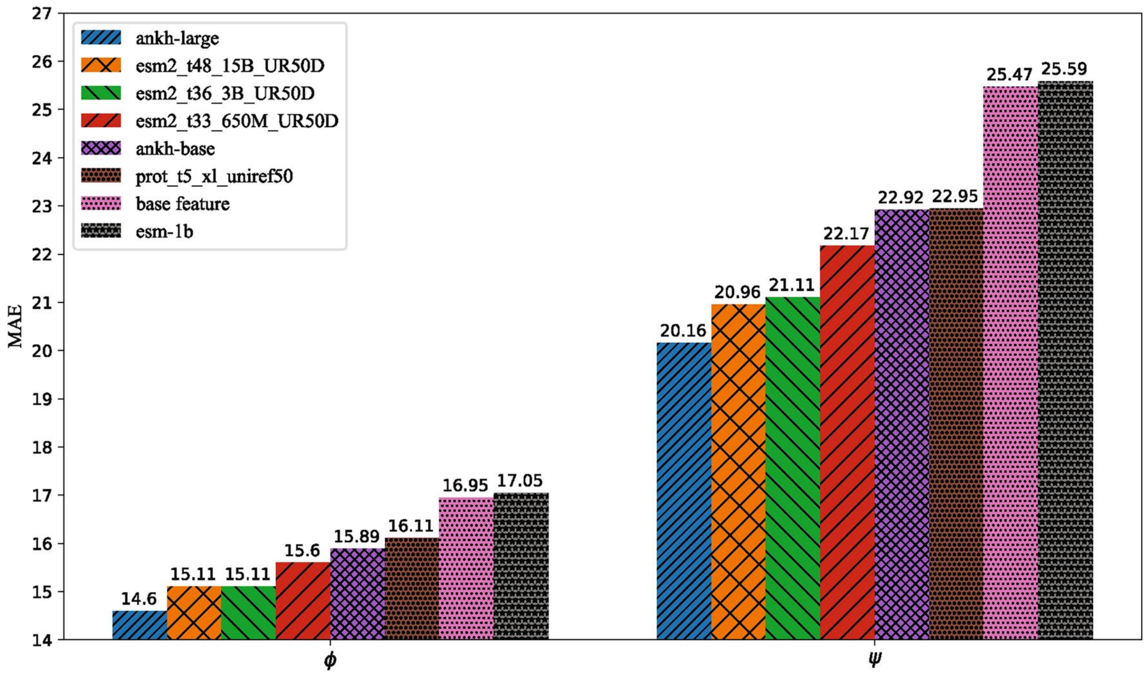

Fig. 4. The mean absolute errors of the proposed method under different feature representations on the test set TEST2016.

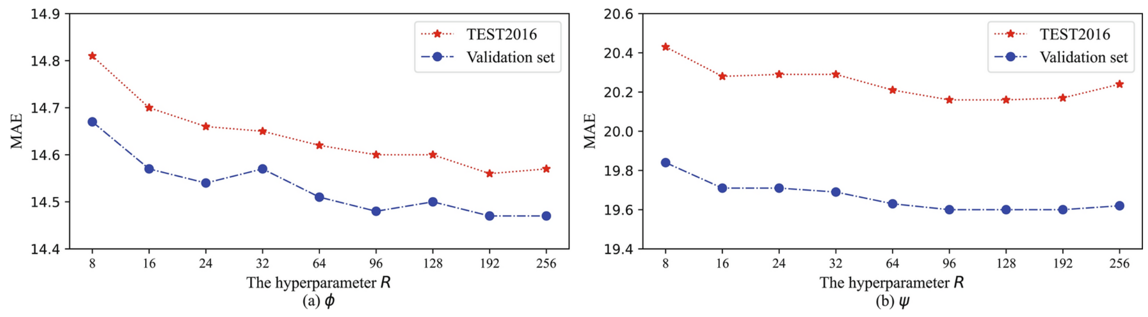

Fig. 5. The mean absolute errors of the proposed method under different R values on the validation set and test set TEST2016.

Impact of the hyperparameter R The hyperparameter R controls the number of parameters in the network layers PHLinear and PHConv1D. The mean absolute errors of the proposed method PHAngle under different R values are shown in Fig. 5. We can see from the figure that the mean absolute errors of the torsion angles ϕ and ψ on the validation set are lowest at $R = 96$. Therefore, we set the default value of the hyperparameter R to 96. In addition, it should be noted that when $R > 96$, further increasing R not only does not guarantee a reduction in the prediction errors of torsion angles, but also increases the model size of the proposed method.

Impact of the channel size To explore the impact of the channel size on the performance of PHAngle, we perform comparative experiments under different C values 512, 768, 1024, 1280, 1536, 1792, and 2048. The mean absolute errors of PHAngle on the validation set and test set TEST2016 are shown in Fig. 6. As can be seen from the figure, when $C >= 1024$, the MAEs of the two torsion angles on both the validation set and the test set fluctuate within the 0.1 interval. This means that PHAngle's torsion angle prediction performance is not sensitive to changes in channel size.

Impact of the network depth The hyperparameter l controls the depth of our backbone network. The prediction errors of the proposed method with varying l values are shown in Fig. 7. As can be seen from the figure, the prediction errors of the torsion angles ϕ and ψ on the validation set reached the minimum simultaneously at $l = 3$. Moreover, further increasing the network depth does not guarantee to reduce the prediction error of the torsion angles. Note that for inference speed considerations, we set the default value of l to 2.

Impact of the loss function In this work, three loss functions, direct prediction loss, indirect prediction loss ($\alpha = 1$) and indirect prediction loss ($\alpha = 2$), can be used to train the proposed method. To investigate their impact on the performance of PHAngle, we perform comparative experiments on the validation set and test

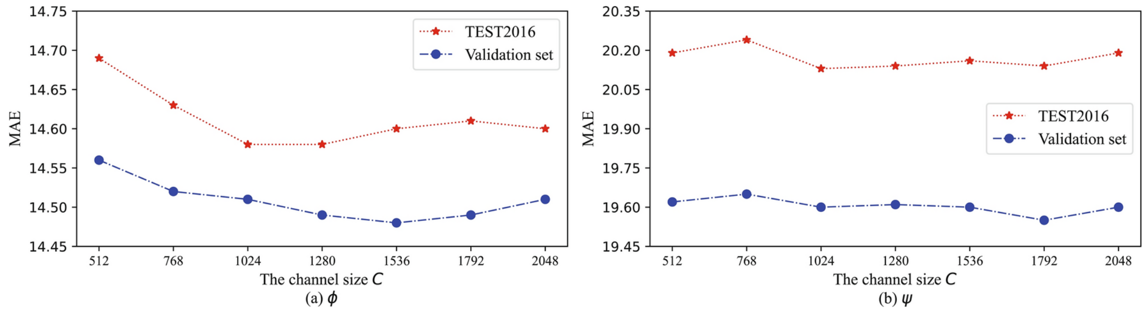

Fig. 6. The mean absolute errors of the proposed method at different channel sizes on the validation set and test set TEST2016.

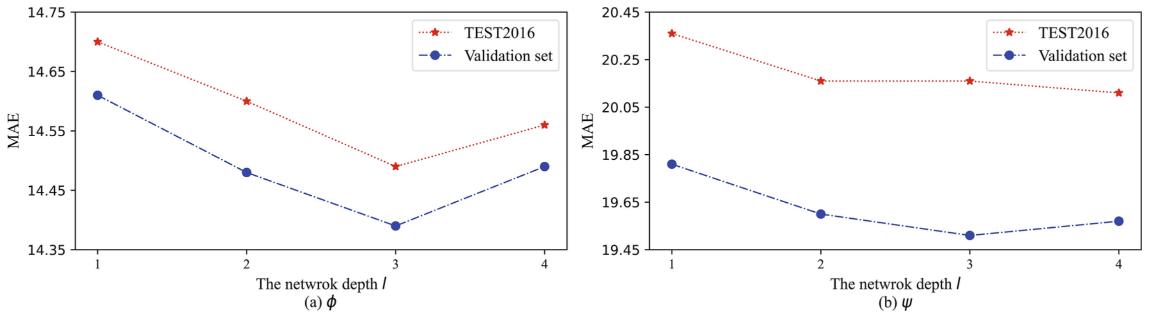

Fig. 7. The mean absolute errors of the proposed method at different network depths on the validation set and test set TEST2016.

Loss function	Validation set		TEST2016	
	MAE(ϕ)	MAE(ψ)	MAE(ϕ)	MAE(ψ)
Direct prediction loss	14.55	19.77	14.64	20.31
Indirect prediction loss ($\alpha = 1$)	**14.30**	**19.49**	**14.39**	**20.15**
Indirect prediction loss ($\alpha = 2$)	14.48	19.60	14.60	20.16

Table 6. The mean absolute errors of the proposed method under different loss functions on the validation set and test set TEST2016. Best results are shown in boldface.

set TEST2016. The prediction results of PHAngle under different loss functions are shown in Table 6. It can be observed that indirect prediction loss ($\alpha = 1$) outperforms direct prediction loss and indirect prediction loss ($\alpha = 2$). In particular, since many existing works such as SPOT-1D[17], DeepRIN[16], and OPUS-TASS[18], adopt the mean square error to define the regression loss of torsion angles, we use the indirect prediction loss ($\alpha = 2$) as the default loss function. Moreover, compared with indirect prediction, the advantage of direct prediction is that there is no need to calculate the sine and cosine values of the angles during training and evaluation.

Conclusion

In this paper, we propose a novel parameter-efficient protein torsion angle predictor, named PHAngle. The PHAngle utilizes parameterized hypercomplex linear and convolutional layers instead of the standard fully connected and convolutional layers to construct the multibranch torsion angle prediction network. This allows it to represent network parameters based on the sum of Kronecker products, thus endowing it with the ability to capture intra- and inter-channel correlations. In particular, PHAngle also significantly reduces the redundancy of network parameters through weight sharing. To eliminate the impact of padded residue positions on nonpadded residue positions during network forward propagation, PHAngle adopts 1-dimensional batch normalization with a mask matrix. Experimental results on the six test sets, TEST2016, TEST2018, TEST2020_HQ, CASP12, CASP13, and CASP-FM, demonstrate that the proposed PHAngle achieves the state-of-the-art performance with the fewest number of parameters compared to the nine existing methods. Moreover, comparison experiments of the proposed method under eight different feature representations show that the embedding feature generated by the pretrained protein language model ankh-large obtain the best torsion angle prediction performance.

Comparison of the three torsion angle loss functions suggests that indirect prediction is superior to direct prediction in terms of prediction error.

Data availability

The source code and datasets are publicly available in the GitHub repository at https://github.com/fengtuan/PHAngle.

Received: 28 April 2024; Accepted: 22 October 2024
Published online: 08 November 2024

References

1. Betancourt, M. R. & Skolnick, J. Local propensities and statistical potentials of backbone dihedral angles in proteins. *J. Mol. Biol.* **342**, 635–649 (2004).
2. Yang, Y., Faraggi, E., Zhao, H. & Zhou, Y. Improving protein fold recognition and template-based modeling by employing probabilistic-based matching between predicted one-dimensional structural properties of query and corresponding native properties of templates. *Bioinformatics* **27**, 2076–2082 (2011).
3. Huang, Y.-M. & Bystroff, C. Improved pairwise alignments of proteins in the twilight zone using local structure predictions. *Bioinformatics* **22**, 413–422 (2005).
4. Neal, S., Berjanskii, M., Zhang, H. & Wishart, D. S. Accurate prediction of protein torsion angles using chemical shifts and sequence homology. *Magn. Reson. Chem.* **44**, S158–S167 (2006).
5. Zimmermann, O. & Hansmann, U. H. E. Support vector machines for prediction of dihedral angle regions. *Bioinformatics* **22**, 3009–3015 (2006).
6. Gao, J., Yang, Y. & Zhou, Y. Grid-based prediction of torsion angle probabilities of protein backbone and its application to discrimination of protein intrinsic disorder regions and selection of model structures. *BMC Bioinf.* **19**, 1–8 (2018).
7. Dor, O. & Zhou, Y. Real-spine: An integrated system of neural networks for real-value prediction of protein structural properties. *Proteins: Struct. Funct. Bioinf.* **68**, 76–81 (2007).
8. Wu, S. & Zhang, Y. Anglor: A composite machine-learning algorithm for protein backbone torsion angle prediction. *PLoS ONE* **3**, e3400 (2008).
9. Faraggi, E., Zhang, T., Yang, Y., Kurgan, L. & Zhou, Y. Spine x: Improving protein secondary structure prediction by multistep learning coupled with prediction of solvent accessible surface area and backbone torsion angles. *J. Comput. Chem.* **33**, 259–267 (2012).
10. Song, J., Tan, H., Wang, M., Webb, G. I. & Akutsu, T. Tangle: Two-level support vector regression approach for protein backbone torsion angle prediction from primary sequences. *PLoS ONE* **7**, e30361 (2012).
11. Heffernan, R. et al. Improving prediction of secondary structure, local backbone angles, and solvent accessible surface area of proteins by iterative deep learning. *Sci. Rep.* **5**, 11476 (2015).
12. Li, H., Hou, J., Adhikari, B., Lyu, Q. & Cheng, J. Deep learning methods for protein torsion angle prediction. *BMC Bioinf.* **18**, 417 (2017).
13. Lyons, J. et al. Predicting backbone C_α angles and dihedrals from protein sequences by stacked sparse auto-encoder deep neural network. *J. Comput. Chem.* **35**, 2040–2046 (2014).
14. Mataeimoghadam, F. et al. Enhancing protein backbone angle prediction by using simpler models of deep neural networks. *Sci. Rep.* **10**, 19430 (2020).
15. Newton, M. A. H., Mataeimoghadam, F., Zaman, R. & Sattar, A. Secondary structure specific simpler prediction models for protein backbone angles. *BMC Bioinf.* **23**, 6 (2022).
16. Fang, C., Shang, Y. & Xu, D. Prediction of protein backbone torsion angles using deep residual inception neural networks. *IEEE/ACM Trans. Comput. Biol. Bioinf.* **16**, 1020–1028 (2019).
17. Hanson, J., Paliwal, K., Litfin, T., Yang, Y. & Zhou, Y. 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**, 2403–2410 (2019).
18. Xu, G., Wang, Q. & Ma, J. Opus-tass: A protein backbone torsion angles and secondary structure predictor based on ensemble neural networks. *Bioinformatics* **36**, 5021–5026 (2020).
19. Xu, Y.-C., ShangGuan, T.-J., Ding, X.-M. & Cheung, N. J. Accurate prediction of protein torsion angles using evolutionary signatures and recurrent neural network. *Sci. Rep.* **11**, 21033 (2021).
20. Elnaggar, A. *et al.* Prottrans: Towards cracking the language of lifes code through self-supervised deep learning and high performance computing. *IEEE Trans. Pattern Anal. Mach. Intell.* **14** (2021).
21. Rives, A. et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl. Acad. Sci.* **118**, e2016239118 (2021).
22. Singh, J., Paliwal, K., Litfin, T., Singh, J. & Zhou, Y. Reaching alignment-profile-based accuracy in predicting protein secondary and tertiary structural properties without alignment. *Sci. Rep.* **12**, 7607 (2022).
23. Høie, M. H. et al. Netsurfp-3.0 Accurate and fast prediction of protein structural features by protein language models and deep learning. *Nucleic Acids Res.* **50**, W510–W515 (2022).
24. Hasan, A., Ahmed, A. Y., Mahbub, S., Rahman, M. S. & Bayzid, M. S. Saint-angle: Self-attention augmented inception-inside-inception network and transfer learning improve protein backbone torsion angle prediction. *Bioinf. Adv.* **3**, vbad042 (2023).
25. Gogoi, C. R., Rahman, A., Saikia, B. & Baruah, A. Protein dihedral angle prediction: The state of the art. *ChemistrySelect* **8**, e202203427 (2023).
26. Zhang, A. *et al.* Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with 1/n parameters. In *International Conference on Learning Representations* (2021).
27. Grassucci, E., Zhang, A. & Comminiello, D. Phnns: Lightweight neural networks via parameterized hypercomplex convolutions. *IEEE Trans. Neural Netw. Learn. Syst.* (2021).
28. Grassucci, E., Sigillo, L., Uncini, A. & Comminiello, D. Hypercomplex image-to-image translation. In *2022 International Joint Conference on Neural Networks (IJCNN)*, 1–8 (2022).
29. Le, T., Bertolini, M., Noé, F. & Clevert, D.-A. Parameterized hypercomplex graph neural networks for graph classification. In *Artificial Neural Networks and Machine Learning*, 204–216 (2021).
30. Panagos, I. I., Sfikas, G. & Nikou, C. Compressing audio visual speech recognition models with parameterized hypercomplex layers. In *Proceedings of the 12th Hellenic Conference on Artificial Intelligence* (2022).
31. Lopez, E., Betello, F., Carmignani, F., Grassucci, E. & Comminiello, D. Attention-map augmentation for hypercomplex breast cancer classification. *Pattern Recognit. Lett.* **182**, 140–146 (2024).
32. Van Loan, C. F. & Pitsianis, N. *Approximation with Kronecker Products* 293–314 (Springer, Netherlands, 1993).
33. Cai, C., Chen, R. & Xiao, H. Kopa: Automated kronecker product approximation. *J. Mach. Learn. Res.* **23**, 1–44 (2022).

34. Yang, W., Hu, Z., Zhou, L. & Jin, Y. Protein secondary structure prediction using a lightweight convolutional network and label distribution aware margin loss. *Knowl.-Based Syst.* **237**, 107771 (2022).
35. Uddin, M. R., Mahbub, S., Rahman, M. S. & Bayzid, M. S. Saint: Self-attention augmented inception-inside-inception network improves protein secondary structure prediction. *Bioinformatics* **36**, 4599–4608 (2020).
36. Elnaggar, A. *et al.* Ankh: Optimized protein language model unlocks general-purpose modelling. arXiv arXiv:2301.06568 (2023).
37. Lin, Z. et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).
38. Hanson, J., Paliwal, K., Litfin, T., Yang, Y. & Zhou, Y. Accurate prediction of protein contact maps by coupling residual two-dimensional bidirectional long short-term memory with convolutional neural networks. *Bioinformatics* **34**, 4039–4045 (2018).

Author contributions
WY designed network architecture and conceived the experiments. SW and LZ prepared the datasets, conducted the experiments and analysed the results. WY and LZ contributed to the original draft preparation, and SW reviewed and edited the manuscript. All authors have read and agreed to the final manuscript.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to L.Z.