GCNdesign Method Summary

Description

This is an overview of an algorithm implemented in the software being available as GCNdesign (https://github.com/ShintaroMinami/GCNdesign). This document has not been peer-reviewed.

Method

The input of the model

The neural network model proposed in this study uses the 3D coordinate of four atoms, N, CA, C, and O, in the input protein main chain structure. Any atoms other than these four were stripped and then added to the pseudo-NH and -CB atoms by assuming the bond distances and bond angles.

- Bond distances of C-C, C=O, and N-H are 1.54, 1.21, and 1.00 Å, respectively.
- Bond angles of N-CA-CB and CB-CA-C are 108.0° and 113.0°, respectively.
- Dihedral angles of C-N-CA-CB and N-C-CA-CB are -124.4° and 121.5°, respectively.

Next, distances among the residue units of the six atoms are calculated to construct an N-nearest neighbor graph, where each residue unit is considered as a node, and the neighborhood relation between the units is considered as an edge. The distance between nodes is defined as the distance between the coordinates of pseudo-CB atoms, and the number of edges per node, N, is set to 20. The dihedral angles (φ, ψ) of each residue were adopted as a feature of the node. Since the angle is a periodic quantity, it is input as a 4-dimensional torus (sinφ, cosφ, sinψ, cosψ). As the edge features, we employed a 6x6 inter-residue distance matrix that considered the six atoms between paired residues. Here, the N-nearest neighbor graph is a directed graph because an edge from residue i to residue j, e_{ij}, and its inverse edge e_{ji} do not necessarily coincide.

Network architecture

The neural network model proposed in this study can be divided into two sub-networks; the encoder network and the prediction network. The encoder network, located upstream, is for computing disentangled and effective features suitable for downstream prediction tasks from the complicated structural features derived from the residues and their surroundings. The downstream prediction network is for predicting the probability of amino acid occurrence at each residue site using the features calculated through the encoder network. 1D convolutional network architecture was employed for the prediction network to reflect the characteristics of the short context of amino acid sequences, which is well known as a concept of the fragment (9 residues).

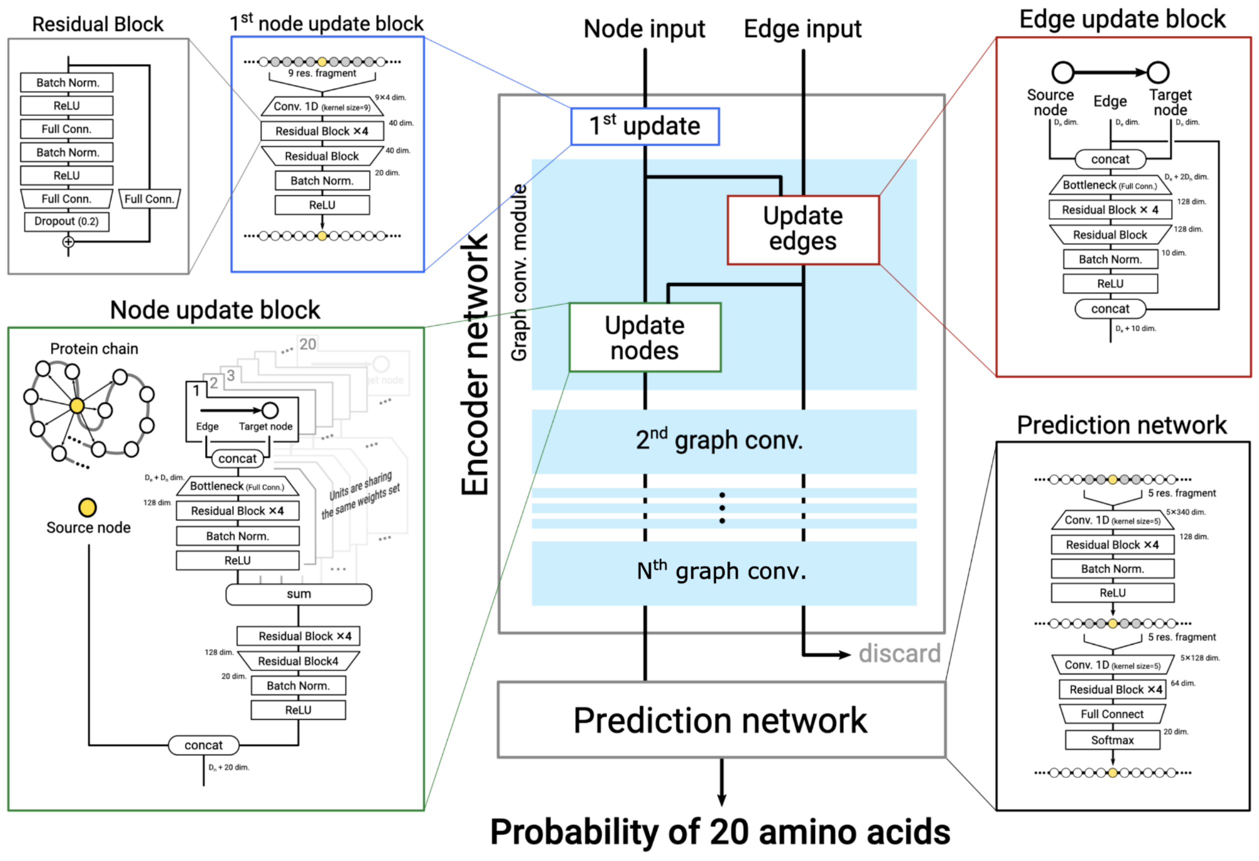

Figure 1: Overview of the network architecture.

Encoder network

The input graph is first subjected to an initial node feature transformation. This operation is computed by repeating the 1D convolution layer and the residual network modules on the sequence of feature vectors arranged along the amino acid sequence. Here a widely-used fragment size (9 residues) was employed as the size of the 1D convolution kernel (Figure1; 1st node update block) without any down-sampling process (stride size=1 and no pooling layer). After this process, a graph convolution operation is applied to iterative, alternating between node transformation and edge transformation operations (Figure 1; Graph Conv. module). The edge and node features are updated in the way of new features being added at each step, as in the DenseNet architecture [1].

In edge update, new features to be added are computed by applying a four-layer residual network module to the information obtained by concatenating the feature vectors of the nodes at both ends of the edge with the feature vector of the edge itself. The edge is updated by concatenating these new features with the feature vector of the original edge (Figure 1; Edge update block).

Updating node features is a bit more complicated than updating edge features (Figure 1; Node update block). First, for each node to be updated, a feature vector is calculated for each of the 20 neighboring nodes. The three kinds of features for each neighboring node, the features of the node of interest, the neighboring node, and the edge between them, are concatenated into one feature vector. The 20 different feature vectors are then aggregated by simple addition and concatenated to the original node's feature vector to update the node information.

The edge update and the node update are combined into a single layer module, which is called the graph convolution layer here [2]. By sequentially applying the graph convolution layer, the vector embedding of the features of nodes and edges is learned, and finally, a feature representation suitable for the downstream prediction problem is obtained. We tested the performance with 3, 4, 5, and 6 graph convolution layers, respectively. The default parameters of the published model employed 5.

Prediction network

Using the feature vectors of each node calculated by the encoder network, the prediction network predicts the probability of amino acid occurrence for each residue site (Figure 1; Prediction network). The prediction module transforms the feature vectors of the nodes, which are arranged in the same way as in the amino acid sequence, into 20-dimensional probability vectors by applying the 1D convolution layer and the 4-layer of residual network module twice, were each with a kernel of five residues including the two residues before and after the node. By applying twice the 1D convolution layer with a kernel size of 5, the predicted amino acid probability of each residue site reflects the information possessed by the fragments of 9-residues in the end.

Training of the DNN model

Dataset

We prepared the dataset based on the structure set introduced by Qi and Zhang [3]. The original training set contained structures with large size of more than 1,000 residues, which could not be calculated due to insufficient GPU memory capacity, so these data were excluded from our training set. The dataset consists of the three subsets; 9,888 structures for training, 500 structures (T500 set) and 50 structures (TS500 set) for testing the performance. These three datasets were created so that the sequence identity is less than 30% for any pairs, including those within and inter datasets. The size of the datasets, counting the total number of residues included, are 2,525,063, 130,960, and 6,860 for the training set, T500 set, and TS50 set, respectively. Since T500 and TS50 sets have been used in a number of previous researches, we can compare the prediction performance with existing methods.

Training

Mini-batch stochastic gradient descent was employed to train the model. The training data cannot be partitioned into arbitrary sizes because the input graph structures should include many edges representing non-local interaction. Therefore, we adopted a method of constructing mini-batches for training by randomly selecting protein chains from the dataset and sequentially adding them into one mini-batch, as long as the total number of residues per mini-batch does not exceed 1,000.

We employed cross-entropy as a loss function for the training of the whole network model. Adam was used as the optimizer, and the learning rate was set to 0.002. The training was continued for 90 epochs and stopped as an early termination. After 80 epochs of learning cycles, the learning rate was multiplied by 0.1, and the remaining 10 epochs were learned for fine-tuning the parameters. Trajectories of loss and per-residue accuracy were monitored during a 5-fold cross-validation (Figure 2). After the hyper-parameters had been fixed, model parameters were trained once using all training data.

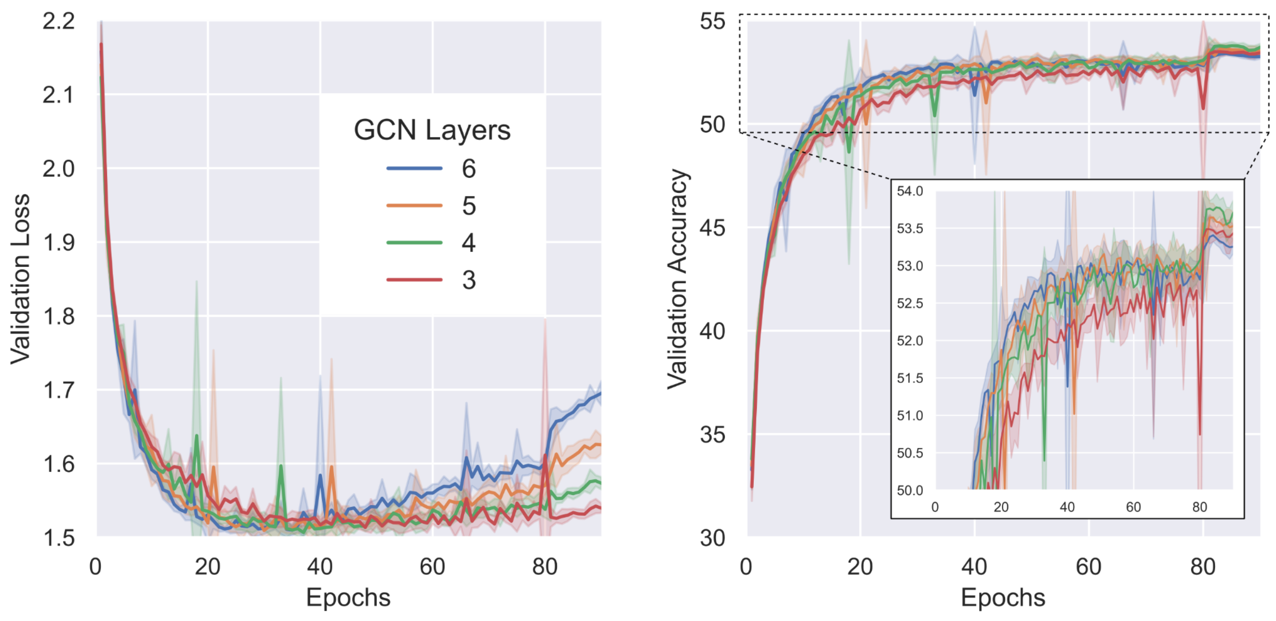

Figure 2: Trajectories for validation loss (left) and validation accuracy (right).

Test

Prediction performance was tested on T500 and TS50 sets. The test results are summarized in Table 1. The accuracy values were obtained from the DenceCPD paper [3] except for the GCNdesign results. In order to compare the performance fairly with existing methods, per-structure accuracy was calculated in the test instead of per-residue accuracy as monitored in the training.

Table 1: Comparison of per-structure accuracy against existing software.

Method	Training (5-fold cross validation)	T500	TS50
GCNdesign (3 layers)	* 53.49 ± 0.12%	53.09%	47.06%
GCNdesign (4 layers)	* 53.78 ± 0.18%	53.38%	47.42%
GCNdesign (5 layers)	* 53.64 ± 0.19%	53.56%	47.61%
GCNdesign (6 layers)	* 53.41 ± 0.08%	52.82%	46.74%
DenseCPD	53.24 ± 0.17%	**55.53%**	**50.71%**
ProDCoNN	NA	42.20%	40.69%
SPROF	NA	40.25%	39.16%
SPIN2	NA	36.60%	33.60%

* Accuracy values for the 5-fold cross-validation training were monitored as per-residue accuracy.
GCNdesign (5 layers) is the one that is used default parameter for the currently available software.

References

1. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. pp. 4700–4708.

2. Kipf TN, Welling M. Semi-Supervised Classification with Graph Convolutional Networks. arXiv [cs.LG]. 2016. Available: http://arxiv.org/abs/1609.02907

3. Qi Y, Zhang JZH. DenseCPD: Improving the Accuracy of Neural-Network-Based Computational Protein Sequence Design with DenseNet. J Chem Inf Model. 2020;60: 1245–1252.