Mathematical Biosciences and Engineering

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

MBE, 20(11): 20188–20212.
DOI: 10.3934/mbe.2023894
Received: 03 August 2023
Revised: 01 October 2023
Accepted: 13 October 2023
Published: 06 November 2023

Research article

Identification of membrane protein types via deep residual hypergraph neural network

Jiyun Shen[1], Yiyi Xia[3,*], Yiming Lu[3], Weizhong Lu[1,2,*], Meiling Qian[1], Hongjie Wu[1], Qiming Fu[1] and Jing Chen[1]

[1] School of Electronic and Information Engineering, Suzhou University of Science and Technology, Suzhou 215009, China

[2] Provincial Key Laboratory for Computer Information Processing Technology, Soochow University, China

[3] Tianping College of Suzhou University of Science and Technology, Suzhou, China

* **Correspondence:** Email: yiyi.xia@usts.edu.cn, luwz@usts.edu.cn.

Abstract: A membrane protein's functions are significantly associated with its type, so it is crucial to identify the types of membrane proteins. Conventional computational methods for identifying the species of membrane proteins tend to ignore two issues: High-order correlation among membrane proteins and the scenarios of multi-modal representations of membrane proteins, which leads to information loss. To tackle those two issues, we proposed a deep residual hypergraph neural network (DRHGNN), which enhances the hypergraph neural network (HGNN) with initial residual and identity mapping in this paper. We carried out extensive experiments on four benchmark datasets of membrane proteins. In the meantime, we compared the DRHGNN with recently developed advanced methods. Experimental results showed the better performance of DRHGNN on the membrane protein classification task on four datasets. Experiments also showed that DRHGNN can handle the over-smoothing issue with the increase of the number of model layers compared with HGNN. The code is available at https://github.com/yunfighting/Identification-of-Membrane-Protein-Types-via-deep-residual-hypergraph-neural-network.

Keywords: hypergraph neural network; initial residual; identity mapping; identification; membrane protein

1. Introduction

The proteins contained in the biological membrane are called membrane proteins, which play a lead role in maintaining many life activities, including but not limited to cell proliferation and differentiation, energy transformation, signal transduction and material transportation. As we know, a membrane protein's functions are significantly associated with its type, so it is important to identify the types of membrane proteins [1]. Membrane proteins can be grouped into eight types [2]: Single-span type 1, single-span type 2, single-span type 3, single-span type 4, multi-span, lipid-anchor, glycosylphosphatidylinositol (GPI)-anchor and peripheral.

There exists many different computational methods that can be used for identifying the types of proteins. Chou and Elrod [3] used the covariant discriminant algorithm (CDA) according to amino acid composition (AAC) to predict membrane protein types. To address the challenge posed by the large number of possible patterns in protein sequences, Chou [4] introduced a pseudo-amino acid composition (PAAC). This composition combines a set of discrete sequence correlation factors with the 20 components of the traditional amino acid composition. Wang et al. [5] utilized the pseudo amino acid composition to incorporate sequence-order effects and introduced spectral analysis for representing the statistical sample of a protein. The weighted support vector machine (SVM) algorithm was applied. Liu et al. [6] introduced the low-frequency Fourier spectrum analysis based on the concept of PAAC, which effectively incorporates sequence patterns into discrete components and enables existing prediction algorithms to be applied directly to protein samples. Chou and Shen [7] developed a two-layer predictor for classifying proteins as membrane or non-membrane. If a protein is classified as membrane, the process continues with a second-layer prediction engine to determine its specific type from eight categories. The predictor stands out for its incorporation of evolutionary information through pseudo position-specific score matrix (Pse-PSSM) vectors and its ensemble classifier consisting of multiple optimized evidence-theoretic K-nearest neighbor (OET-KNN) classifiers. Rezaei et al. [8] classified membrane proteins by applying wavelet analysis to their sequences and extracting informative features. These features were normalized and used as input for a cascaded model, which aimed to mitigate bias caused by differences in membrane protein class sizes in the dataset. Wang et al. [9] utilized the dipeptide composition (DC) method to represent proteins as high-dimensional feature vectors. They introduced the neighborhood preserving embedding (NPE) algorithm for linear dimensionality reduction and to extract essential features from the high-dimensional DC space. The reduced low-dimensional features were then employed with the K-nearest neighbor (K-NN) classifier to accurately classify membrane protein types. Hayat and Khan [10] integrated composite protein sequence features (CPSR) with the PAAC to classify membrane protein. They further proposed using split amino acid composition (SAAC) and ensemble classification [11] and still further fused position specific scoring matrix (PSSM) and SAAC [12] to classify membrane protein. Chen and Li [13] introduced a novel computational classifier designed for the prediction of membrane protein types using protein sequences. The classifier was constructed based on a collection of one-versus-one SVMs and incorporated various sequence attributes. Han et al. [14] integrated amino acid classifications and physicochemical properties in PAAC and used a two-stage multiclass SVM to classify membrane protein. Wan et al. [15] retrieved the associated gene ontology (GO) information of a query membrane protein by searching a compact GO-term database with its homologous accession number. Subsequently, they employed a multi-label elastic net (EN) classifier to classify the membrane protein based on this information. Lu et al. [16] used a dynamic deep network architecture that was based on lifelong learning for the classification of membrane protein. Wang et al. [17] introduced a new support bio-sequence machine, which used SVM for protein classification.

In conclusion, most of the above models used different computational methods to represent membrane proteins and then used classification algorithms to identify membrane protein types. Most of the models mentioned above have varied types of feature input formats, which are shown in Table 1.

Table 1. Varied types of feature input formats of different methods.

Methods	**Input form**
MemType-2L [7]	Pseudo-PSSM (PsePSSM)
predMPT [13]	Pseudo amino acid composition
CDA [3]	Amino acid composition
CDA and PseAA [4]	Pseudo amino acid composition
Fourier spectrum [6]	Pseudo amino acid composition
Weighted SVM [5]	Pseudo amino acid composition
Wavelet and cascade neural network [8]	Hydropathy signal
NPE [9]	Dimension-reduced vector(50-D) by NPE
CPSR [10]	Pseudo amino acid composition
Two-stage SVM [14]	Pseudo amino acid composition

However, it is noted that more than two proteins are linked by non-covalent interactions [18, 19] in real practice, and the representation of proteins is multi-modal. Traditional computational methods for identifying membrane protein types tend to ignore those two issues, which leads to information loss since the high-order correlation among membrane proteins and the scenarios of multi-modal representations of membrane proteins are ignored.

To tackle those problems, in this paper we use a deep residual hypergraph neural network (DRHGNN) [20] to further learn about the representations of membrane proteins and to eventually achieve accurate identification of membrane proteins' types.

First, each membrane protein is represented by the extracted features. Here, five feature extraction methods are employed based on the PSSM of membrane protein sequence [2], including average blocks (AvBlock), discrete cosine transform (DCT), discrete wavelet transform (DWT), histogram of oriented gradient (HOG) and PsePSSM. Five types of features are extracted accordingly. Second, each feature type generates a hypergraph G represented by an incidence matrix H modeling complex high-order correlation. Five types of features and corresponding incidence matrix H are concatenated, respectively, which overcomes the scenarios of multi-modal representations of membrane proteins. Lastly, concatenated features and fused incidence matrix are input into a DRHGNN to classify the various types of membrane proteins. To assess the performance of DRHGNN, we perform tests on membrane proteins' four distinct datasets. In the task of membrane proteins classification, the model achieves better performance.

2. Materials and methods

In order to extract features of membrane proteins, we employ AvBlock, DCT, DWT, HOG and PsePSSM [2] to achieve feature extraction based on membrane protein sequence's PSSM. Each type of PSSM-based feature is used to generate a hypergraph that can be represented by an incidence matrix H, then five types of features and their corresponding H are concatenated, respectively, and both are fed into a DRHGNN [20–22] to identify the types of membrane proteins. Figure 1 depicts the schematic diagram.

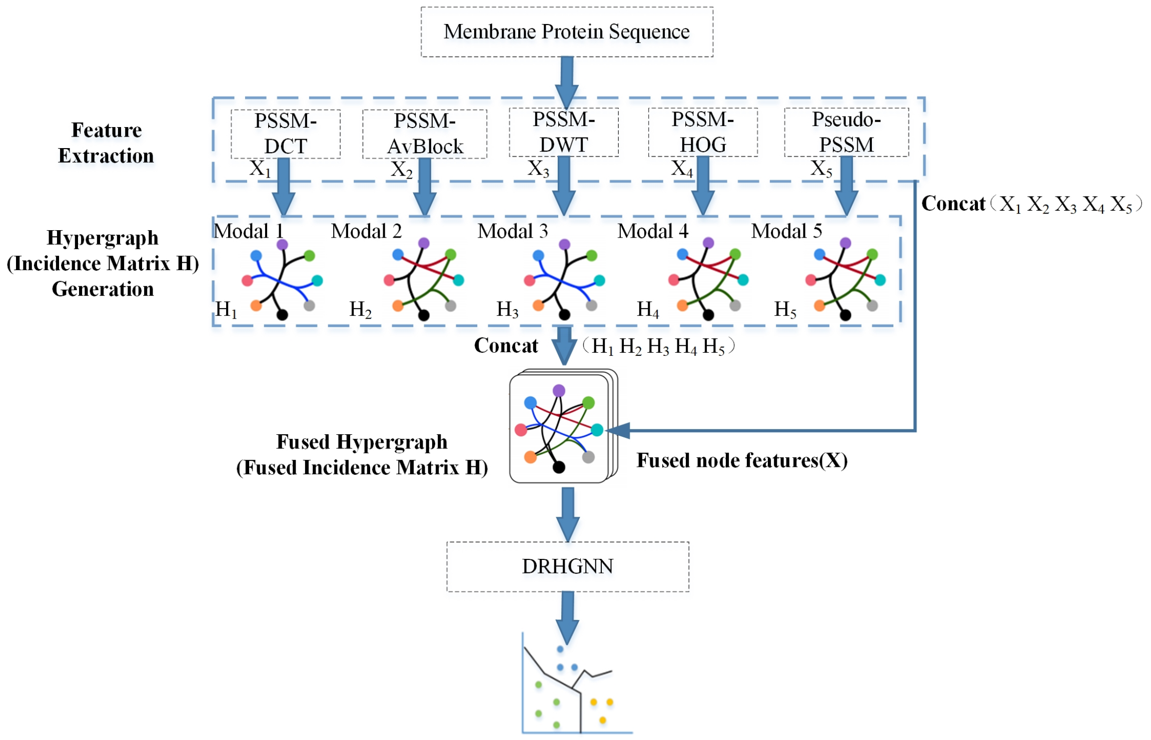


Figure 1. The schematic diagram of our proposed method.

2.1. *Data set*

We judge the performance of DRHGNN on the classification of membrane proteins based on four datasets, namely, Dataset 1, Dataset 2, Dataset 3 and Dataset 4.

Dataset 1 is directly sourced from Chou's work [7], where protein sequences are sourced from the Swiss-Prot [23] database. Chou and Shen [7] employed a percentage distribution method to randomly assign the protein sequences into both the training set and the testing set. This was done to ensure a balanced number of sequences between the two sets. Dataset 1 consists of 7582 membrane proteins from eight types and the same training/testing split as [7], where 3,249 membrane proteins are employed for training, with the remaining 4,333 employed for testing.

Dataset 2 was created by removing redundant and highly similar sequences from Dataset 1. This resulted in a curated dataset with reduced homology, specifically ensuring that no pair of proteins shared a sequence identity greater than 40%. The training set of Dataset 2 was obtained by removing redundant sequences from Dataset 1's training set. Similarly, the testing set of Dataset 2 was prepared by eliminating redundant sequences and those with high sequence identity to the training set. Dataset 2 consists of 4594 membrane proteins from eight types and the same training/testing split as [13], where 2288 membrane proteins are employed for training, with the remaining 2306 membrane proteins employed for testing.

To update and expand the datasets, Chen and Li [13] created Dataset 3 through the following steps. Initially, membrane protein sequences were obtained from the Swiss-Prot [23] database using the "protein subcellular localization" annotation. Stringent exclusion criteria was applied to ensure dataset quality: 1) Exclusion of fragmented proteins or those shorter than 50 amino acid residues; 2) removal of proteins with non-experimental qualifiers or multiple topologies in their annotations; 3) elimination of homologous sequences with a sequence identity greater than 40% using clustering database at high identity with tolerance (CD-hit) [24]. Subsequently, the sequences were categorized into their respective membrane protein types based on topology annotations. To generate the training and testing sets, a random assignment was performed employing the above-mentioned percentage distribution method. Consequently, Dataset 3 was created, providing an updated and expanded dataset of membrane protein sequences characterized by enhanced quality and classification. Dataset 3 consists of 6677 membrane proteins from eight types and the same training/testing split as [13], where 3,073 membrane proteins are employed for training and 3604 for testing.

Dataset 4 is directly sourced from Chou's work [3], where protein sequences are sourced from the Swiss-Prot [23] database. The training and testing sets were obtained after protein sequences were screened with three procedures. Dataset 4 consists of 4684 membrane proteins from five types and the same training/testing split as [3], where 2059 membrane proteins are used for training and 2625 membrane proteins are employed for testing. Table 2 outlines the details of the datasets.

Table 2. The scale of training and testing samples in four different membrane proteins' datasets.

Specific types	**Dataset 1**		**Dataset 2**		**Dataset 3**		**Dataset 4**	
	Train	Test	Train	Test	Train	Test	Train	Test
Single-span type 1	610	444	388	223	561	245	435	478
Single-span type 2	312	78	218	39	316	7	152	180
Single-span type 3	24	6	19	6	32	9	–	–
Single-span type 4	44	12	35	10	65	17	–	–
Multi-span type 5	1316	3265	936	1673	1119	2478	1311	1867
Lipid-anchor type 6	151	38	98	26	142	36	51	14
GPI-anchor type 7	182	46	122	24	164	41	110	86
Peripheral type 8	610	444	472	305	674	699	–	–
Overall	3249	4333	2288	2306	3073	3604	2059	2625

We use the same membrane protein features as [2], which are extracted with five methods based on the PSSM of membrane proteins.

2.2. PSSM

The PSSM is a widely used tool in the field of bioinformatics for capturing evolutionary information encoded within membrane protein sequences. It is generated through multiple sequence alignment and database searching methods, such as position-specific iterated BLAST (PSIBLAST) program [25], to identify conserved residues and their positional probabilities.

The evolutionary information obtained from the PSSM is preserved within a matrix of size R × 20 (R rows and 20 columns), presented as follows:

$$PSSM = \begin{pmatrix} p_{1,1} & \cdots & p_{1,j} & \cdots & p_{1,20} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ p_{i,1} & \cdots & p_{i,j} & \cdots & p_{i,20} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ p_{R,1} & \cdots & p_{R,j} & \cdots & p_{R,20} \end{pmatrix}. \tag{2.1}$$

The numbers 1–20 denote one of the 20 different amino acids. R denotes the length of the membrane protein sequence.The element $p_{i,j}$ is calculated as follows:

$$p_{i,j} = \sum_{k=1}^{20} \omega(i,k) \times D(k,j); i = 1, \ldots, L, j = 1, \ldots, 20. \tag{2.2}$$

$\omega(i,k)$ represents the frequency of the k-th amino acid type at position i, and D(k, j) denotes the value derived from Dayhoff's mutation matrix (substitution matrix) for the k-th and j-th amino acid types. The utilization of these variables in the equation aims to incorporate amino acid frequency information and substitution probabilities.

2.3. AvBlock

AvBlock refers to a statistical measure employed in professional scientific research to analyze data sequences. Nowadays, AvBlock is a widely adopted approach for constructing matrix descriptors to represent protein sequences [26]. AvBlock is calculated by dividing the total length of a sequence by the average length of its individual consecutive blocks. Here, the PSSM matrix is partitioned into 20 blocks along the rows. Subsequently, each block is transformed into a feature vector of dimensionality 20 for the PSSM matrix.

2.4. DCT

The DCT [27] is a mathematical transform widely used in signal and image processing. Here, we employ a two-dimensional DCT (2D-DCT) for compressing the PSSM of proteins. The mathematical definition for the 2D-DCT is

$$F_{PSSM-DCT} = \alpha_i \alpha_j \textstyle\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} PSSM(m,n) \cos \frac{\pi(2m+1)i}{2M} \cos \frac{\pi(2n+1)j}{2N} \tag{2.3}$$

$$\alpha_i = \begin{cases} \sqrt{1/M}, & i = 0 \\ \sqrt{2/M}, & 1 \leq i \leq M-1 \end{cases} \tag{2.4}$$

$$\alpha_j = \begin{cases} \sqrt{1/N}, & j = 0 \\ \sqrt{2/N}, & 1 \leq j \leq N-1, \end{cases} \tag{2.5}$$

where $0 < i < M$ and $0 < j < N$.

2.5. DWT

The DWT has been utilized to extract informative features from protein amino acid sequences, as initially introduced by Nanni et al. [28]. Here, we applied a 4-level DWT to preprocess the PSSM matrix. At each level we compute both the approximate and detailed coefficients for each column. We extract essential statistical features such as maximum, minimum, mean and standard deviation from both the approximate and detailed coefficients. Additionally, we capture the first five discrete cosine coefficients exclusively from the approximate coefficients. Therefore, for each of the 20 column dimensions, a total of 4 + 4 + 5 features are obtained at each level.

2.6. HOG

The HOG is a feature descriptor used in computer vision and image processing for object detection and recognition. Here, we propose a method to reduce redundancy in protein data using the HOG algorithm. We consider the PSSM as an image-like matrix representation. First, we compute the horizontal and vertical gradients of the PSSM to obtain the gradient magnitude and direction matrices. These matrices are then partitioned into 25 sub-matrices that incorporate both the gradient magnitude and direction information. Subsequently, we generate 10 distinct histogram channels for each sub-matrix based on its gradient direction. This approach effectively reduces redundancy by providing a compact representation of the protein data while preserving important spatial information.

2.7. PsePSSM

The PsePSSM is a commonly utilized matrix descriptor in protein research [7]. It is specifically designed to preserve the essential information contained in the PSSM by considering the incorporation of PAAC. The PsePSSM descriptor is formulated as follows:

$$F_{PsePSSM} = \begin{cases} \frac{1}{N}\sum_{i=1}^{N} p'_{i,j};\ j = 1,\dots,20 \\ \frac{1}{N-lag}\sum_{i=1}^{N-lag}\left(p'_{i,j} - p'_{i+lag,j}\right)^2;\ j = 1,\dots,20,\ \text{lag} = 1,\dots,30, \end{cases} \tag{2.6}$$

where *lag* refers to the distance between a residue and its neighboring residues. The formula of $p'_{i,j}$ is

$$p'_{i,j} = \frac{p_{i,j} - \frac{1}{20}\sum_{m=1}^{20} p_{i,m}}{\sqrt{\frac{1}{20}\sum_{n=1}^{20}\left(p_{i,n} - \frac{1}{20}\sum_{m=1}^{20} p_{i,m}\right)^2}}, \tag{2.7}$$

where $p'_{i,j}$ refers to the normalized version of $p_{i,j}$.

2.8. DRHGNN

2.8.1. Hypergraph learning statement

In a basic graph, the samples are depicted as vertexes, and two connected vertexes are joined by an edge [29, 30]. However, the data structure in practical applications may go beyond pair connections and may even be multi-modal. Accordingly, the hypergraph was proposed. Unlike the simple graph, a hypergraph comprises a vertex set and one or more hyperedge set(s) composed of two or more vertexes, as shown in Figure 2. A hypergraph is represented by G = (V, E, W), where V represents a vertex set, and E represents a hyperedge set. W, a diagonal matrix of edge weights, assigns weights to each hyperedge. The incidence matrix H, where H is the $|V| \times |E|$ incidence matrix with entries defined as

$$h(v, e) = \begin{cases} 1, \text{ if } v \in \text{e} \\ 0, \text{ if } v \notin \text{e}, \end{cases} \tag{2.8}$$

is used to denote the hypergraph.

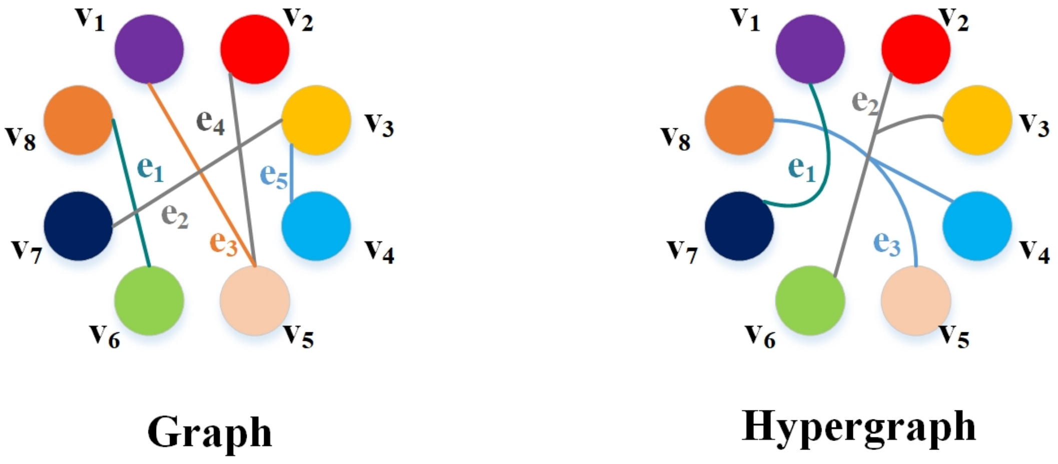


Figure 2. The comparison between graph and hypergraph.

Here, we could take the membrane protein classification task on the hypergraph because more than two proteins are linked by non-covalent interactions [18, 19]. $X = [\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{N}]^\mathrm{T}$ can represent the features of N membrane proteins data. The hyperedge is constructed using the Euclidean distance, which calculates the distance expressed with $\mathrm{d}\left(\mathrm{x}_\mathrm{i}, \mathrm{x}_\mathrm{j}\right)$ between two features. In the hyperedge construction, each vertex represents a membrane protein, and then one central vertex and its K neighbors represent each hyperedge. As a result, N hyperedges containing K+1 vertexes are generated. Here, more specifically, each time we select one vertex in the dataset as the centroid, we use K nearest neighbors in the selected feature space to generate one hyperedge, which includes the centroid itself, as illustrated in Figure 3. Thus, a hypergraph with N hyperedges is constructed with a single-modal representation of membrane proteins. The hypergraph is denoted by an incidence matrix $\mathrm{H} \in \mathrm{R}^{\mathrm{NxN}}$, with Nx(K+1) nonzero entries denoting $v \in e$ while the others equal zero.

In the case of multi-modal representations of membrane proteins, each incidence matrix H_i is constructed according to each modality membrane representation. After all the incidence matrix H_i have been generated, these H_i can be concatenated to generate the incidence matrix H of a multi-modality hypergraph. Thus, a hypergraph is constructed with multi-modal representations of membrane proteins shown in Figure 3, so it is noted that the flexibility of hypergraph generation has great expansibility toward multi-modal features.

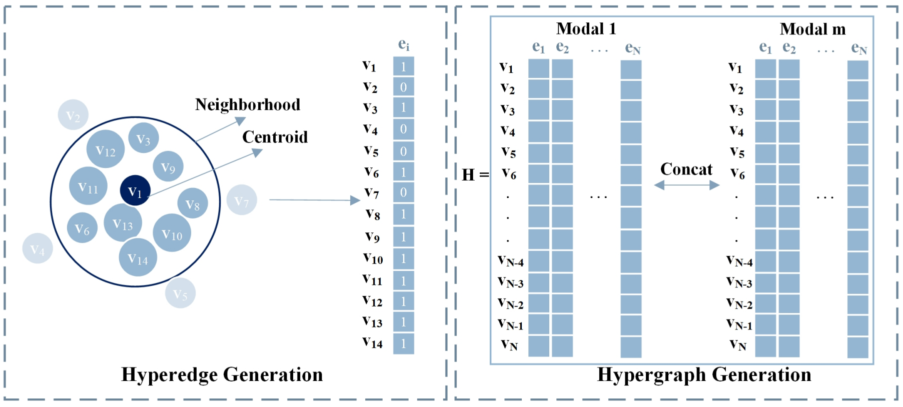


Figure 3. The schematic diagram of hyperedge generation and hypergraph generation.

2.8.2. Hypergraph convolution

Feng et al. [31] first proposed the HGNN. They built a hyperedge convolution layer whose formulation is

$$X^{(1+1)} = \sigma\left(D_v^{-1/2} H W D_e^{-1} H^T D_v^{-1/2} X^{(1)} \Theta^{(1)}\right), \tag{2.9}$$

where $X^{(l)} \in R^{NxC}$ represents the hypergraph's signal at the *lth* layer with N nodes and C dimensional features, W is regarded as the weight of all hyperedges and $W = \operatorname{diag}(w_1, \ldots, w_N)$.$\Theta^{(l)} \in R^{C_1 x C_2}$ represents the parameter that is learned during the training process at the *lth* layer. σ represents the nonlinear activation function. D_v is the vertex degrees' diagonal matrix, while D_e is the edge degrees' diagonal matrix [30].

We define hypergraph Laplacian $\widetilde{H} = D_v^{-1/2} H W D_e^{-1} H^T D_v^{-1/2}$, then a hyperedge convolution layer is formulated as $X^{(l+1)} = \sigma\left(\widetilde{H} X^{(l)} \Theta^{(l)}\right)$.

A hyperedge convolution layer achieves node-edge-node transform, which can refine a better representation of nodes and extract the high-order correlation from a hypergraph more efficiently.

2.8.3. Residual hypergraph convolution

Feng et al. [31] used two hyperedge convolution layers and then used the softmax function to obtain predicted labels. However, the performance of HGNN drops as the number of layers increases because of the over-smoothing issue.

To resolve the issue of over-smoothing, Huang et al. [20] and Chen et al. [22] used two simple and effective techniques, Initial residual and identity mapping, based on their shallow model. Inspired by their method, we upgrade the HGNN by introducing initial residual and identity mapping to prevent over-smoothing and enjoy accuracy increase from increased depth.

- Initial residual

  Chen et al. [22] constructed a connection to the initial representation $X^{(0)}$ to relieve the over-smoothing problem. The initial residual connection guarantees that each node's final representation retains at least a proportion of the input feature regardless of how many layers we stack.

Gasteiger et al. [32] proposed approximate personalized propagation of neural predictions (APPNP), which employed a linear combination between different layers to the initial residual connection and gathered information from multi-hop neighbors instead of expanding the number of neural network layers by separating feature transformation and propagation. Formally, APPNP's model is defined as

$$X^{(l+1)} = \sigma\left(\left((1-\alpha_l)\widetilde{H}X^{(l)} + \alpha_l X^{(0)}\right)\Theta^{(l)}\right). \tag{2.10}$$

In practice, we can set $\alpha_l = 0.1 \; or \; 0.2$.

- Identity mapping

  However, APPNP remains a shallow model; thus, the initial residual alone cannot extend HGNN to a deep model. To resolve this issue, Chen et al. [22] added an identity matrix I_N to the weight matrix $\Theta^{(l)}$ according to the idea in ResNet of identity mapping, which ensures the DRHGNN model performs at least as well as its shallow version does.

  Finally, a residual enhanced hyperedge convolution layer is formulated as

$$X^{(l+1)} = \sigma\left(\left((1-\alpha_l)\widetilde{H}X^{(l)} + \alpha_l X^{(0)}\right)\cdot\left((1-\beta_l)\, I_n + \beta_l\Theta^{(l)}\right)\right). \tag{2.11}$$

In practice, we set $\beta_l = \frac{\lambda}{l}$, where λ is a hyperparameter.

2.8.4. DRHGNN analysis

Figure 4 illustrates the detail of the DRHGNN. Those multi-types of node features and corresponding incidence matrix H modeling complex high-order correlation are concatenated, respectively, which overcomes the scenarios of multi-modal representations of membrane proteins. Then, concatenated features and incidence matrix are fed into DRHGNN to get nodes output labels and eventually achieve classification task. As detailed in the section mentioned above, we can build a residual enhanced hypergraph convolution layer, then we naively stack multiple residual hypergraph convolution blocks to tackle the problem of over-smoothing in HGNN and enjoy an accuracy increase. Additional linear transforms are incorporated into the model's first and last layer, and the residual hypergraph convolutions are utilized for information propagation. The deep embeddings are finally used for classification tasks.

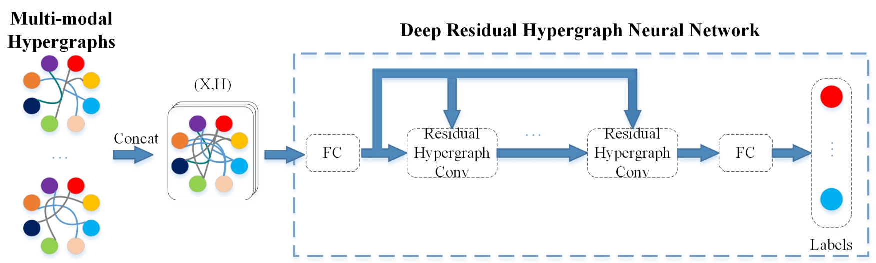


Figure 4. The DRHGNN framework. FC represents a fully connected layer.

3. Results

3.1. Hyperparameter settings based on experience

The DRHGNN has numerous hyperparameters. Instead of comparing all the possible hyperparameters, which usually takes several days, we used empirically based hyperparameters, which are shown in Table 3.

We implemented our model with Pytorch [33].

The baseline results were replicated by their release codes, with hyperparameters adhering to the respective papers.

Table 3. Hyperparameters used in this study.

Hyperparameters	**Setting**
Epoch	2000
Learning rate	0.001
Hidden layer	128
Dropout rate	0.5
Activation function	ReLU+Softmax
Optimizer	Adam
Loss function	Cross-entropy loss function
α	0.1
λ	2.5

3.2. Metrics

We conducted accuracy calculations for predicting every type of membrane protein. We used accuracy (ACC), which measures the ratio of correctly predicted proteins to the total number of proteins in a specified dataset, to assess the performance of our model. The specific formula is

$$ACC = \frac{n}{N}, \tag{3.1}$$

where n stands for the number of proteins that are correctly predicted in a specified dataset, and N stands for the total number of proteins present in the dataset.

In order to further evaluate the performance of models, we also incorporated F1-score and Mathew's correlation coefficient (MCC) as evaluation metrics.

The F1-score is a useful metric for addressing the issue of imbalanced datasets, which is composed of precision and recall. Precision refers to the ratio of the number of correctly predicted samples to the total number of samples predicted as positive, while recall refers to the ratio of the number of correctly predicted samples to the total number of actual positive samples. The best value of F1-score is 1, while the worst value is 0. Their specific formulas are

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{(\text{Precision} + \text{Recall})} \tag{3.2}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.3}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{3.4}$$

where TP, TN, FP and FN are true positive, true negative, false positive and false negative, respectively.

In order to comprehensively evaluate the F1-scores of multiple classes, we employed the macro average of F1-score, which aggregates the F1-score for different classes by taking their average with equal weights assigned to all classes.

MCC is widely acknowledged as a superior performance metric for the classification of imbalanced data. It is defined within the range of $[-1, 1]$, where a value of 1 indicates that the classifier accurately predicts all positive instances and negative instances, while a value of -1 signifies that the classifier incorrectly predicts all instances. The specific formula is

$$MCC(i) = \frac{TP \times TN - FP \times FN}{\sqrt{[TP + FP][TP + FN][TN + FP][TN + FN]}}. \tag{3.5}$$

The overall MCC for all categories is computed by averaging the MCC values of individual categories.

3.3. The selection of K value when constructing the hypergraph

The selection of K neighbors plays a vital role in the construction process of the hyperedge, as it has a significant impact on the model's performance. The selection of K value is performed by training the model with different K values and evaluating its performance. The optimal K value is determined based on the performance metric obtained from the validation set. We performed K value experiments on four datasets using DRHGNN. The performance metric is macro average of the F1-score. As observed from Table 4, each dataset achieves the best experimental result at different K values, specifically $K = 8$, 10, 12 and 2 respectively. Therefore, when conducting experiments on the four datasets, we selected K values in sequence as 8, 10, 12 and 2.

Table 4. The performance of DRHGNN with different K values on four datasets. The best result for each dataset is bolded.

Datasets	**Metric**	**K**						
		2	4	6	8	10	12	14
Dataset 1	F1-score	0.68282	0.74181	0.73623	**0.75769**	0.75697	0.75656	0.74846
Dataset 2	F1-score	0.65046	0.65052	0.64668	0.67813	**0.68646**	0.68438	0.67955
Dataset 3	F1-score	0.57224	0.56307	0.59114	0.59234	0.58799	**0.59317**	0.59105
Dataset 4	F1-score	**0.95651**	0.94911	0.94645	0.95541	0.94468	0.94590	0.94242

3.4. Performance comparison of DRHGNN and HGNN with different layers

The performance of DRHGNN against HGNN with different layers on four datasets is reported in Table 5. Columns 4–9 show the ACC, macro average of the F1-score between DRHGNN and HGNN with different layers on four datasets. For better comparison, we presented the results in

Figure 5. By analyzing Table 5 and Figure 5, we can observe two points: 1) DRHGNN achieves much better performance than HGNN on four datasets with accuracy gains of 3.738, 3.903, 4.106, 1.028%, respectively, and with a macro average of F1-score gains of 15.306, 11.843, 13.887, 3.591%, respectively, with their optimal layer. 2) The residual enhanced model (DRHGNN) has stable performance, while the performance of HGNN deteriorates as the number of layers increases. The potential reason for HGNN's performance degradation with increasing layer depth is that the model may suffer from an over-smoothing issue. The performance of DRHGNN persistently improves and achieves the best accuracy on four datasets at layer 4, 8, 4 and 8, respectively, and the best macro average of the F1-score on four datasets at layer 4, 8, 4 and 16, respectively.

Table 5. Comparison of the ACC, macro average of F1-score between DRHGNN and HGNN with different depths on four datasets. The best result of methods for each dataset is bolded.

Datasets	**Methods**	**Metrics**	**Layers**					
			2	4	8	16	32	64
Dataset 1	HGNN	ACC	**0.90492**	0.90122	0.88114	0.70688	0.75352	0.75352
		F1-score	**0.60463**	0.46066	0.32800	0.27766	0.12858	0.11633
	DRHGNN	ACC	0.92799	**0.94230**	0.93492	0.93561	0.93330	0.93215
		F1-score	0.68801	**0.75769**	0.73092	0.72435	0.75289	0.70492
Dataset 2	HGNN	ACC	**0.86904**	0.84996	0.80833	0.81049	0.72550	0.72550
		F1-score	**0.55921**	0.42989	0.28404	0.28703	0.12723	0.11472
	DRHGNN	ACC	0.89072	0.89809	**0.90807**	0.89592	0.89245	0.89549
		F1-score	0.66307	0.67684	**0.67764**	0.66848	0.65073	0.64934
Dataset 3	HGNN	ACC	**0.85322**	0.85294	0.81104	0.80549	0.68757	0.68785
		F1-score	**0.45430**	0.37403	0.27717	0.26733	0.12236	0.11141
	DRHGNN	ACC	0.88957	**0.89428**	0.89179	0.88929	0.88873	0.88652
		F1-score	0.59160	**0.59317**	0.57857	0.57942	0.57635	0.56952
Dataset 4	HGNN	ACC	0.96343	**0.97562**	0.96762	0.93486	0.71124	0.71124
		F1-score	0.89026	**0.93080**	0.90231	0.75977	0.16625	0.16625
	DRHGNN	ACC	0.98590	0.98400	**0.98590**	0.98514	0.98133	0.98019
		F1-score	0.95951	0.95651	0.96116	**0.96671**	0.95890	0.95858

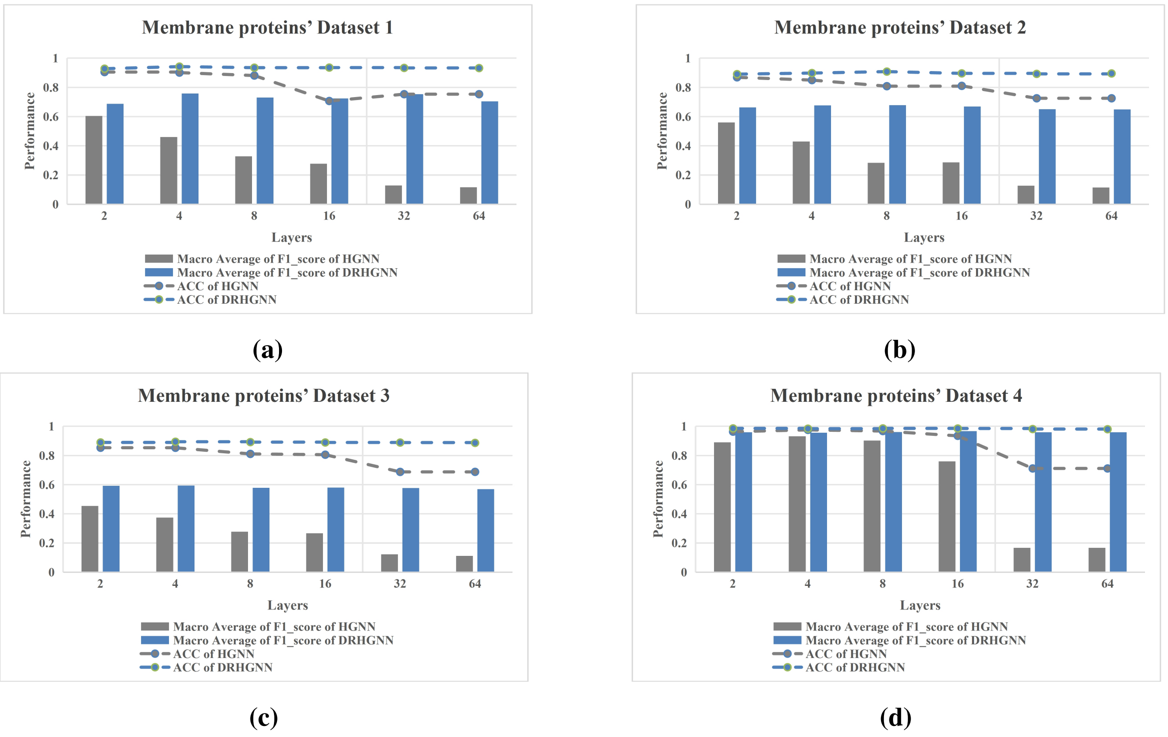


Figure 5. The performance comparison of DRHGNN and HGNN with different layers on membrane protein classification task. (a) The performance comparison of DRHGNN and HGNN on Dataset 1; (b) The performance comparison of DRHGNN and HGNN on Dataset 2; (c) The performance comparison of DRHGNN and HGNN on Dataset 3; (d) The performance comparison of DRHGNN and HGNN on Dataset 4.

3.5. Performance comparison with multiple recently developed advanced methods

The summaries of classification accuracy results of DRHGNN with multiple recently developed advanced methods are shown in Tables 6–9. Tables 6–Table 8 present a comparison of the accuracy of each type of membrane protein and the overall accuracy across all membrane proteins for Dataset 1, Dataset 2, and Dataset 3 using different methods. As Tables 6–8 show, the accuracy of each type of membrane protein obtained using our method is generally higher than those achieved by other methods, and the overall accuracy is also superior to that of other methods. More specifically, compared with the MemType-2L [7] and hypergraph neural network [34] on Dataset 1, DRHGNN achieves overall accuracy gains of 2.63 and 3.738%, respectively. Compared with the MemType-2L [7] and hypergraph neural network [34] on Dataset 2, DRHGNN achieves overall accuracy gains of 5.507 and 3.903%, respectively. Compared with the MemType-2L [7] and hypergraph neural network [34] on Dataset 3, DRHGNN achieves overall accuracy gains of 11.128 and 4.106%, respectively. Furthermore, within these three datasets, the fifth type of membrane protein exhibits the highest accuracy compared to other types. This can potentially be attributed to the significantly larger number of samples available for the fifth type of membrane protein in these datasets. Table 9 presents a comparison of the overall accuracy between our proposed method and other methods on Dataset 4. As Table 9 shows, our method achieved the best performance among all the compared methods. More specifically, compared with CPSR [10] and two-stage SVM [14] on Dataset 4, DRHGNN achieves overall accuracy gains of 3.314 and 1.814%, respectively. Those results demonstrate the superior performance of DRHGNN on the membrane protein classification task. The detailed performance of DRHGNN on four datasets is shown in Table 10.

Table 6. Comparison of the ACC between DRHGNN and multiple recent state of the art methods on Dataset 1. The best result among methods is bolded.

Membrane protein types	**Metric**	**MemType-2L [7]**	**Hypergraph neural network [34]**	**DRHGNN**
Single-span type 1	ACC	0.86900	0.90090	**0.93468**
Single-span type 2	ACC	0.70500	0.34615	**0.80769**
Single-span type 3	ACC	0.33333	0.16667	**0.33333**
Single-span type 4	ACC	0.66667	0.33333	**0.66667**
Multi-span	ACC	0.95000	0.93813	**0.96478**
Lipid-anchor	ACC	0.42100	0.26316	**0.55263**
GPI-anchor	ACC	0.76100	0.63043	**0.84783**
Peripheral	ACC	0.82200	**0.87162**	0.86712
Overall	ACC	0.91600	0.90492	**0.94230**

Table 7. Comparison of the ACC between DRHGNN and multiple recent state of the art methods on Dataset 2. The best result among methods is bolded.

Membrane protein types	**Metric**	**MemType-2L [7]**	**Hypergraph neural network [34]**	**DRHGNN**
Single-span type 1	ACC	0.76700	0.77130	**0.85650**
Single-span type 2	ACC	**0.66700**	0.23077	0.64103
Single-span type 3	ACC	**0.33333**	0.16667,	0.16667
Single-span type 4	ACC	**0.70000**	0.40000	0.60000
Multi-span	ACC	0.91400	0.93843	**0.94979**
Lipid-anchor	ACC	0.23100	0.23077	**0.26923**
GPI-anchor	ACC	0.70800	0.58333	**0.75000**
Peripheral	ACC	0.68200	0.74754	**0.84262**
Overall	ACC	0.85300	0.86904	**0.90807**

Table 8. Comparison of the ACC between DRHGNN and multiple recent state of the art methods on Dataset 3. The best result among methods is bolded.

Membrane protein types	Metric	MemType-2L [7]	Hypergraph neural network [34]	DRHGNN
Single-span type 1	ACC	0.69000	0.60816	**0.77959**
Single-span type 2	ACC	**0.58200**	0.16456	0.40506
Single-span type 3	ACC	**0.55600**	0.11111	0.11111
Single-span type 4	ACC	**0.52900**	0.17647	0.29412
Multi-span	ACC	0.90700	0.93826	**0.95642**
Lipid-anchor	ACC	0.33333	0.08333	**0.36111**
GPI-anchor	ACC	0.65900	0.29268	**0.73171**
Peripheral	ACC	0.43900	0.81402	**0.83119**
Overall	ACC	0.78300	0.85322	**0.89428**

Table 9. Comparison of the ACC between DRHGNN and multiple recently developed advanced methods on Dataset 4. The best result among methods is bolded.

Methods	Input form	ACC
CDA [3]	Amino acid composition	0.79400
CDA and PseAA [4]	Pseudo amino acid composition	0.87500
Fourier spectrum [6]	Pseudo amino acid composition	0.87000
Weighted SVM [5]	Pseudo amino acid composition	0.90300
Wavelet and cascade neural network [8]	Hydropathy signal	0.91400
NPE [9]	Dimension-reduced vector(50-D) by NPE	0.90100
CPSR [10]	Pseudo amino acid composition	0.95200
Two-stage SVM [14]	Pseudo amino acid composition	0.96700
DRHGNN	PSSM-DCT+ PSSM-AvBlock+ PSSM-DWT+ PSSM-HOG+ PsePSSM	**0.98514**

Table 10. The detailed performance of DRHGNN on four datasets.

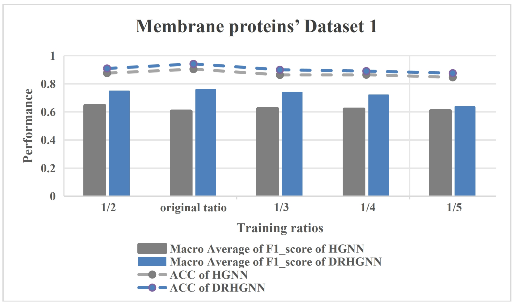
Membrane protein types	**Metrics**	**Dataset 1**	**Dataset 2**	**Dataset 3**	**Dataset 4**
	ACC	0.93468	0.85650	0.77959	0.98536
Single-span type 1	F1-score	0.92325	0.82773	0.80160	0.98312
	MCC	0.91445	0.84770	0.78271	0.98590
	ACC	0.80769	0.64103	0.40506	0.98889
Single-span type 2	F1-score	0.74556	0.63636	0.44330	0.93931
	MCC	0.74282	0.63485	0.42195	0.93091
	ACC	0.33333	0.16667	0.11111	–
Single-span type 3	F1-score	0.44444	0.36364	0.15385	–
	MCC	0.47091	0.23432	0.16530	–
	ACC	0.66667	0.60000	0.29412	–
Single-span type 4	F1-score	0.76190	0.75000	0.41667	–
	MCC	0.76927	0.77392	0.36325	–
	ACC	0.96478	0.94979	0.95642	0.98500
Multi-span	F1-score	0.97448	0.95333	0.95528	0.98952
	MCC	0.90025	0.84768	0.86025	0.96609
	ACC	0.55263	0.26923	0.36111	1.00000
Lipid-anchor	F1-score	0.58333	0.41860	0.34783	0.93333
	MCC	0.58077	0.34900	0.30896	0.93506
	ACC	0.84783	0.75000	0.73171	0.97674
GPI-anchor	F1-score	0.80412	0.69565	0.81579	0.98824
	MCC	0.80300	0.71752	0.80484	0.98188
	ACC	0.86712	0.84262	0.83119	–
Peripheral	F1-score	0.82441	0.77578	0.81104	–
	MCC	0.80448	0.75171	0.77898	–
	ACC	0.94230	0.90807	0.89428	0.98514
Overall	F1-score	0.75769	0.67764	0.59317	0.96671
	MCC	0.86341	0.79716	0.78434	0.96788

3.6. Stability analysis

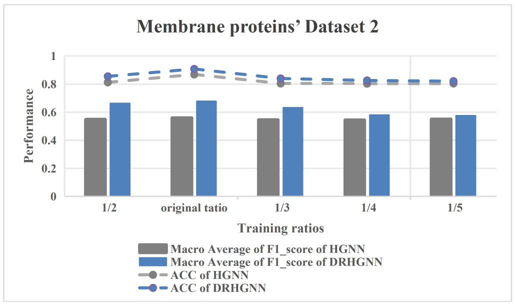
To further analyze the stability of DRHGNN compared to HGNN, we conducted an analysis by adjusting the training rate. All experiments were carried out with five different training rates followed by five distinct seeds. We then recorded the best results using the optimal number of layers in each experiment. Table 11 and Figure 6 show that DRHGNN consistently performs better than HGNN across all training rates, with around 1.5 to 5% overall accuracy enhancements and around 3.591 to 15.306% macro average of F1-score enhancements. This demonstrates the stability of DRHGNN performing better than HGNN with different ratios. In the meanwhile, DRHGNN shows stability, especially in small training rates and shows its better performance around original training rate.

Table 11. Summaries of the ACC, macro average of F1-score of DRHGNN and HGNN with different training ratios.

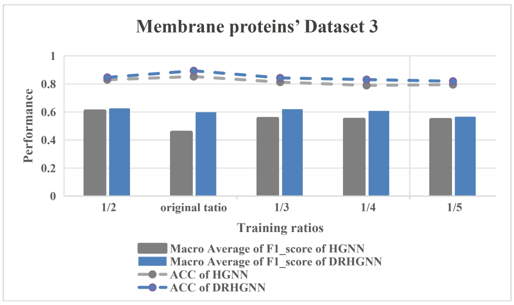
Datasets	**Methods (optimal layer)**	**Metrics**	**Training ratios**				
			1/2	Original ratio	1/3	1/4	1/5
Dataset 1	HGNN (2)	ACC	0.87658	**0.90492**	0.86373	0.86518	0.84660
		F1-score	0.64517	**0.60463**	0.62267	0.61952	0.60766
	DRHGNN (4)	ACC	0.90981	**0.94230**	0.90032	0.89102	0.87626
		F1-score	0.74753	**0.75769**	0.73867	0.71947	0.63653
Dataset 2	HGNN (2)	ACC	0.81261	**0.86904**	0.80463	0.80336	0.80479
		F1-score	0.54794	**0.55921**	0.54456	0.54314	0.54980
	DRHGNN (8)	ACC	0.85522	**0.90807**	0.84018	0.82599	0.82110
		F1-score	0.66241	**0.67764**	0.63211	0.58003	0.57609
Dataset 3	HGNN (2)	ACC	0.83029	**0.85322**	0.81324	0.78986	0.79566
		F1-score	0.60610	**0.45430**	0.55233	0.54659	0.54499
	DRHGNN (4)	ACC	0.84705	**0.89428**	0.84332	0.83137	0.81961
		F1-score	0.62085	**0.59317**	0.61548	0.60365	0.56172
Dataset 4	HGNN (4)	ACC	0.96073	**0.97562**	0.94912	0.93739	0.93091
		F1-score	0.86638	**0.93080**	0.85814	0.81250	0.82102
	DRHGNN (16)	ACC	0.96628	**0.98514**	0.95200	0.95304	0.93305
		F1-score	0.90371	**0.96671**	0.87237	0.87466	0.84134



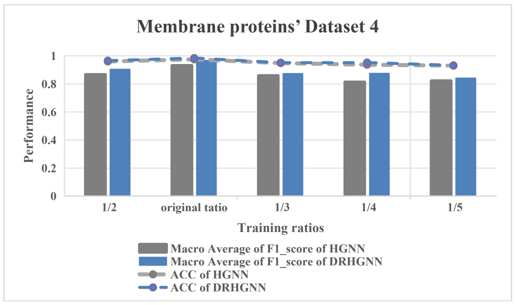
(a)



(b)



(c)



(d)

Figure 6. Stability analysis. The performance of DRHGNN and HGNN with different training ratios on membrane protein classification task. (a) The performance on Dataset 1; (b) The performance on Dataset 2; (c) The performance on Dataset 3; (d) The performance on Dataset 4.

3.7. Ablation study

We conducted an ablation study on initial residual and identity mapping. In Table 12, columns 4-9 show the accuracy and the macro average of the F1-score of four methods with different depths of the network layers on the four datasets. As Table 12 and Figure 7 show, HGNN using identify mapping can mitigate the problem of over-smoothing a little, and HGNN using initial residual can reduce the over-smoothing problem greatly. Meanwhile, adopting initial residual and identity mapping together can significantly improve performance while effectively reducing the over-smoothing problem. Furthermore, we found that the experimental results of HGNN adopting initial residual and identity mapping together and HGNN using initial residual are very close. However, HGNN adopting both outperforms in terms of accuracy and the macro average of the F1-score and reaches the best result faster than just adopting the initial residual.

4. Conclusions

This study proposed a DRHGNN enhanced with initial residual and identity mapping based on HGNN to further learn the representations of membrane proteins for identifying the types of membrane proteins.

First, the extracted features generated with five methods represented each membrane protein. Second, each incidence matrix H_i was constructed according to each modality membrane protein representation. Lastly, those multi-modals of membrane protein features and corresponding H_i were concatenated, respectively, and both were fed into the DRHGNN for the membrane protein classification task.

Table 12. Ablation study on initial residual and identity mapping. The best result of methods for each dataset is bolded.

Datasets	**Methods**	**Metrics**	**Layers** 2	4	8	16	32	64
Dataset 1	HGNN	ACC	**0.90492**	0.90122	0.88114	0.70688	0.75352	0.75352
		F1-score	**0.60463**	0.46066	0.32800	0.27766	0.12858	0.11633
	HGNN with initial residual	ACC	0.93284	0.93353	**0.93607**	0.93515	0.94023	0.93515
		F1-score	0.73563	0.70930	0.73086	0.74404	**0.75760**	0.74492
	HGNN with identity mapping	ACC	**0.92361**	0.90238	0.87399	0.83383	0.82206	0.78791
		F1-score	**0.66054**	0.53962	0.36386	0.28012	0.26553	0.23277
	DRHGNN	ACC	0.92799	**0.94230**	0.93492	0.93561	0.93330	0.93215
		F1-score	0.68801	**0.75769**	0.73092	0.72435	0.75289	0.70492
Dataset 2	HGNN	ACC	**0.86904**	0.84996	0.80833	0.81049	0.72550	0.72550
		F1-score	**0.55921**	0.42989	0.28404	0.28703	0.12723	0.11472
	HGNN with initial residual	ACC	0.88942	0.89462	0.89029	0.89289	**0.89592**	0.89636
		F1-score	0.65802	0.66908	0.63511	0.65379	**0.67675**	0.64619
	HGNN with identity mapping	ACC	**0.88682**	0.84389	0.80703	0.79618	0.77450	0.75412
		F1-score	**0.62149**	0.39402	0.29337	0.26617	0.23825	0.22974
	DRHGNN	ACC	0.89072	0.89809	**0.90807**	0.89592	0.89245	0.89549
		F1-score	0.66307	0.67684	**0.67764**	0.66848	0.65073	0.64934
Dataset 3	HGNN	ACC	**0.85322**	0.85294	0.81104	0.80549	0.68757	0.68785
		F1-score	**0.45430**	0.37403	0.27717	0.26733	0.12236	0.11141
	HGNN with initial residual	ACC	0.88707	**0.89095**	0.88957	0.88485	0.88235	0.88430
		F1-score	0.57256	**0.59146**	0.57517	0.58479	0.56690	0.58825
	HGNN with identity mapping	ACC	**0.86820**	0.84212	0.80327	0.80105	0.77913	0.77691
		F1-score	**0.50946**	0.37121	0.26852	0.25792	0.19333	0.19148
	DRHGNN	ACC	0.88957	**0.89428**	0.89179	0.88929	0.88873	0.88652
		F1-score	0.59160	**0.59317**	0.57857	0.57942	0.57635	0.56952
Dataset 4	HGNN	ACC	0.96343	**0.97562**	0.96762	0.93486	0.71124	0.71124
		F1-score	0.89026	**0.93080**	0.90231	0.75977	0.16625	0.16625
	HGNN with initial residual	ACC	0.98514	**0.98590**	0.98286	**0.98590**	0.98438	0.98438
		F1-score	0.95924	**0.96382**	0.96120	0.96776	0.95881	0.95886
	HGNN with identity mapping	ACC	**0.98590**	0.97943	0.96800	0.95276	0.94057	0.90400
		F1-score	**0.96033**	0.95568	0.93180	0.85132	0.75033	0.68475
	DRHGNN	ACC	**0.98590**	0.98400	**0.98590**	0.98514	0.98133	0.98019
		F1-score	0.95951	0.95651	0.96116	**0.96671**	0.95890	0.95858

In those extensive experiments on membrane protein classification task, our method achieved a much better performance on four datasets.

DRHGNN resolves the following issues: The high-order correlation among membrane proteins and the scenarios of multi-modal representations of membrane proteins. In the meantime, DRHGNN can handle the over-smoothing issue as the number of model layers increases compared with HGNN.

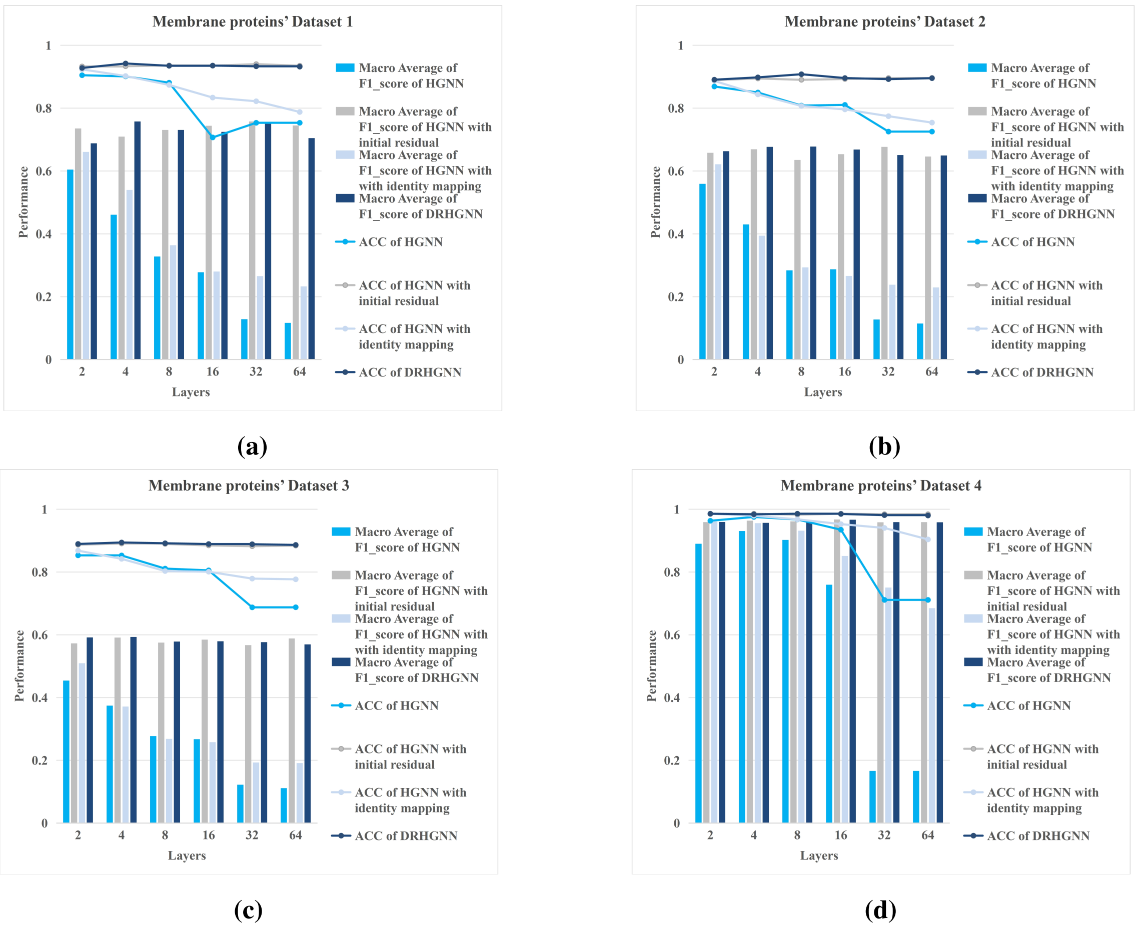


Figure 7. Ablation study on initial residual and identity mapping. The performance comparison of DRHGNN, HGNN, HGNN with initial residual, HGNN with identity mapping with different layers on membrane protein classification task. (a) The performance comparison on Dataset 1; (b) The performance comparison on Dataset 2; (c) The performance comparison on Dataset 3; (d) The performance comparison on Dataset 4.

However, we found three areas for improvement while doing experiments. One is that DRHGNN is quite sensitive to different datasets. Specifically, the performance of Dataset 4 is better than the performance of other datasets. The overall quantity of Dataset 4, the partitioning of training and testing sets on Dataset 4 and the distribution of a certain membrane protein class on Dataset 4 differ from the other datasets, which may significantly influence the training process and generalization capabilities of the model. Another is that we ignored the modification of hyperedge following with adjusted feature embedding in different layers. The model is still worth enhancing. The third one is that the hyperedges were constructed based on feature similarity, which may not directly represent physical interactions between the membrane proteins. Our approach should be considered as an approximation rather than a direct representation of interactions.

The main challenge for future research is to resolve three issues: DRHGNN's sensitivity to different datasets, modification of hyperedge following with adjusted feature embedding in different layers and capturing physical interactions among membrane proteins accurately.

In the meantime, the progress in interaction prediction research across diverse fields of computational biology holds great promise for gaining valuable insights into genetic markers and ncRNAs associated with membrane protein types, such as the prediction of miRNA-IncRNA interactions using a method based on the graph convolutional neural (GCN) network and the conditional random field (CRF) [35], gene function and protein association (GFPA) that extracts reliable associations between gene function and cell surface proteins from single-cell multimodal data [36], prediction of lncRNA-miRNA association using a network distance analysis model [37], prediction of the potential associations of disease-related metabolites using GCN with graph attention network [38], predicting Human ether-a-go-go-related gene (hERG) blockers using molecular fingerprints and graph attention mechanism [39] and predicting the potential associations between metabolites and diseases based on autoencoder and nonnegative matrix factorization [40]. These will also be our future research direction.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This paper is supported by the National Natural Science Foundation of China (62372318, 61902272, 62073231, 62176175, 61876217, 61902271), the National Research Project (2020YFC2006602), the Provincial Key Laboratory for Computer Information Processing Technology, Soochow University (KJS2166) and the Opening Topic Fund of Big Data Intelligent Engineering Laboratory of Jiangsu Province (SDGC2157).

Conflict of interest

All authors declare no conflicts of interest in this paper.

Data availability statement

The code is available at https://github.com/yunfighting/Identification-of-Membrane-Protein-Types-via-deep-residual-hypergraph-neural-network. The dataset presented in this study is available on request from the corresponding author.

References

1. X. Zhang, L. Chen, Prediction of membrane protein types by fusing protein-protein interaction and protein sequence information, *Biochim. Biophys Acta Proteins Proteom.*, **1868** (2020), 140524. https://doi.org/10.1016/j.bbapap.2020.140524

2. H. Wang, Y. Ding, J. Tang, F. Guo, Identification of membrane protein types via multivariate information fusion with Hilbert-Schmidt independence criterion, *Neurocomputing*, **383** (2020), 257–269. https://doi.org/10.1016/j.neucom.2019.11.103

3. K. Chou, D. W. Elrod, Prediction of membrane protein types and subcellular locations, *Proteins*, **34** (1999), 137–153.

4. K. Chou, Prediction of protein cellular attributes using pseudo-amino acid composition, *Proteins*, **43** (2001), 246–255. https://doi.org/10.1002/prot.1035

5. M. Wang, J. Yang, G. Liu, Z. Xu, K. Chou, Weighted-support vector machines for predicting membrane protein types based on pseudo-amino acid composition, *Protein Eng. Des. Sel.*, **17** (2004), 509–516. https://doi.org/10.1093/protein/gzh061

6. H. Liu, M. Wang, K. Chou, Low-frequency Fourier spectrum for predicting membrane protein types, *Biochem. Biophys. Res. Commun.*, **336** (2005), 737–739. https://doi.org/10.1016/j.bbrc.2005.08.160

7. K. Chou, H. Shen, MemType-2L: A web server for predicting membrane proteins and their types by incorporating evolution information through Pse-PSSM, *Biochem. Biophys. Res. Commun.*, **360** (2007), 339–345. https://doi.org/10.1016/j.bbrc.2007.06.027

8. M. A. Rezaei, P. Abdolmaleki, Z. Karami, E. B. Asadabadi, M. A. Sherafat, H. Abrishami-Moghaddam, et al., Prediction of membrane protein types by means of wavelet analysis and cascaded neural networks, *J. Theor. Biol.*, **254** (2008), 817–820. https://doi.org/10.1016/j.jtbi.2008.07.012

9. L. Wang, Z. Yuan, X. Chen, Z. Zhou, The prediction of membrane protein types with NPE, *IEICE Electron. Express*, **7** (2010), 397–402. https://doi.org/10.1587/elex.7.397

10. M. Hayat, A. Khan, Predicting membrane protein types by fusing composite protein sequence features into pseudo amino acid composition, *J. Theor. Biol.*, **271** (2011), 10–17. https://doi.org/10.1016/j.jtbi.2010.11.017

11. M. Hayat, A. Khan, M. Yeasin, Prediction of membrane proteins using split amino acid and ensemble classification, *Amino Acids*, **42** (2012), 2447–2460. https://doi.org/10.1007/s00726-011-1053-5

12. M. Hayat, A. Khan, MemHyb: predicting membrane protein types by hybridizing SAAC and PSSM, *J. Theor. Biol.*, **292** (2012), 93–102. https://doi.org/10.1016/j.jtbi.2011.09.026

13. Y. Chen, K. Li, Predicting membrane protein types by incorporating protein topology, domains, signal peptides, and physicochemical properties into the general form of Chou's pseudo amino acid composition, *J. Theor. Biol.*, **318** (2013), 1–12. https://doi.org/10.1016/j.jtbi.2012.10.033

14. G. Han, Z. Yu, V. Anh, A two-stage SVM method to predict membrane protein types by incorporating amino acid classifications and physicochemical properties into a general form of Chou's PseAAC, *J. Theor. Biol.*, **344** (2014), 31–39. https://doi.org/10.1016/j.jtbi.2013.11.017

15. S. Wan, M. Mak, S. Kung, Mem-mEN: Predicting multi-functional types of membrane proteins by interpretable elastic nets, *IEEE/ACM Trans. Comput. Biol. Bioinf.*, **13** (2016), 706–718. https://doi.org/10.1109/TCBB.2015.2474407

16. W. Lu, J. Shen, Y. Zhang, H. Wu, Y. Qian, X. Chen, et al., Identifying membrane protein types based on lifelong learning with dynamically scalable networks, *Front. Genet.*, **12** (2022), 2787. https://doi.org/10.3389/fgene.2021.834488

17. Y. Wang, Y. Zhai, Y. Ding, Q. Zou, SBSM-Pro: Support bio-sequence machine for proteins, preprint, arXiv:2308.10275.

18. J. B. Pereira-Leal, E. D. Levy, S. A. Teichmann, The origins and evolution of functional modules: lessons from protein complexes, *Philos. Trans. R. Soc. B Biol. Sci.*, **361** (2006), 507–517. https://doi.org/10.1098/rstb.2005.1807

19. E. D. Levy, J. B. Pereira-Leal, C. Chothia, S. A. Teichmann, 3D complex: A structural classification of protein complexes, *PLoS Comput. Biol.*, **2** (2006), 155. https://doi.org/10.1371/journal.pcbi.0020155

20. J. Huang, X. Huang, J. Yang, Residual enhanced multi-hypergraph neural network, in *2021 IEEE International Conference on Image Processing (ICIP)*, (2021), 3657–3661. https://doi.org/10.1109/ICIP42928.2021.9506153

21. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2016), 770–778. https://doi.org/10.1109/CVPR.2016.90

22. M. Chen, Z. Wei, Z. Huang, B. Ding, Y. Li, Simple and deep graph convolutional networks, in *Proceedings of the 37th International Conference on Machine Learning*, (2020), 1725–1735.

23. B. Boeckmann, A. Bairoch, R. Apweiler, M. Blatter, A. Estreicher, E. Gasteiger, et al., The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003, *Nucleic Acids Res.*, **31** (2003), 365–370. https://doi.org/10.1093/nar/gkg095

24. W. Li, A. Godzik, Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences, *Bioinformatics*, **22** (2006), 1658–1659. https://doi.org/10.1093/bioinformatics/btl158

25. S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, et al., Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, *Nucleic Acids Res.*, **25** (1997), 3389–3402. https://doi.org/10.1093/nar/25.17.3389

26. J. C. Jeong, X. Lin, X. Chen, On position-specific scoring matrix for protein function prediction, *IEEE/ACM Trans. Comput. Biol. Bioinf.*, **8** (2011), 308–315. https://doi.org/10.1109/TCBB.2010.93

27. N. Ahmed, T. Natarajan, K. R. Rao, Discrete cosine transform, *IEEE Trans. Comput.*, **23** (1974), 90–93. https://doi.org/10.1109/T-C.1974.223784

28. L. Nanni, S. Brahnam, A. Lumini, Wavelet images and Chou's pseudo amino acid composition for protein classification, *Amino Acids*, **43** (2012), 657–665. https://doi.org/10.1007/s00726-011-1114-9

29. B. Schölkopf, J. Platt, T. Hofmann, Learning with hypergraphs: Clustering, classification, and embedding, in *Advances in Neural Information Processing Systems 19*, MIT Press, (2007), 1601–1608.

30. Y. Gao, M. Wang, D. Tao, R. Ji, Q. Dai, 3-D object retrieval and recognition with hypergraph analysis, *IEEE Trans. Image Process.*, **21** (2012), 4290–4303. https://doi.org/10.1109/TIP.2012.2199502

31. Y. Feng, H. You, Z. Zhang, R. Ji, Y. Gao, Hypergraph neural networks, in *The Thirty-Third AAAI Conference on Artificial Intelligence*, **33** (2019), 3558–3565. https://doi.org/10.1609/aaai.v33i01.33013558

32. J. Gasteiger, A. Bojchevski, S. Günnemann, Predict then propagate: Graph neural networks meet personalized pageRank, in *Seventh International Conference on Learning Representations*, (2019).

33. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, et al., PyTorch: An imperative style, high-performance deep learning library, in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, (2019), 8026–8037.

34. W. Lu, M. Qian, Y. Zhang, H. Wu, Y. Ding, J. Shen, et al., Identification of membrane protein types based using hypergraph neural network, *Curr. Bioinf.*, **18** (2023), 346–358. http://doi.org/10.2174/1574893618666230224143726

35. W. Wang, L. Zhang, J. Sun, Q. Zhao, J. Shuai, Predicting the potential human lncRNA–miRNA interactions based on graph convolution network with conditional random field, *Briefings Bioinf.*, **23** (2022), 463. https://doi.org/10.1093/bib/bbac463

36. H. Hu, Z. Feng, H. Lin, J. Cheng, J. Lyu, Y. Zhang, et al., Gene function and cell surface protein association analysis based on single-cell multiomics data, *Comput. Biol. Med.*, **157** (2023), 106733. https://doi.org/10.1016/j.compbiomed.2023.106733

37. L. Zhang, P. Yang, H. Feng, Q. Zhao, H. Liu, Using network distance analysis to predict lncRNA–miRNA interactions, *Interdiscip. Sci.*, **13** (2021), 535-545. https://doi.org/10.1007/s12539-021-00458-z

38. F. Sun, J. Sun, Q. Zhao, A deep learning method for predicting metabolite–disease associations via graph neural network, *Briefings Bioinf.*, **23** (2022), 266. https://doi.org/10.1093/bib/bbac266

39. T. Wang, J. Sun, Q. Zhao, Investigating cardiotoxicity related with hERG channel blockers using molecular fingerprints and graph attention mechanism, *Comput. Biol. Med.*, **153** (2023), 106464. https://doi.org/10.1016/j.compbiomed.2022.106464

40. H. Gao, J. Sun, Y. Wang, Y. Lu, L. Liu, Q. Zhao, et al., Predicting metabolite–disease associations based on auto-encoder and non-negative matrix factorization, *Briefings Bioinf.*, **24** (2023), 259. https://doi.org/10.1093/bib/bbad259

AIMS Press