PAPER • OPEN ACCESS

SICGNN: structurally informed convolutional graph neural networks for protein classification

To cite this article: YongHyun Lee *et al* 2024 *Mach. Learn.: Sci. Technol.* **5** 045072

MACHINE LEARNING
Science and Technology

CrossMark

OPEN ACCESS

RECEIVED
2 August 2024

REVISED
11 November 2024

ACCEPTED FOR PUBLICATION
26 November 2024

PUBLISHED
23 December 2024

PAPER

SICGNN: structurally informed convolutional graph neural networks for protein classification

YongHyun Lee[1,2], Eunchan Kim[3], Jiwoong Choi[4,5,*] and Changhyun Lee[2,6,*]

1 Department of Computer Science and Engineering, Seoul National University, Seoul, Republic of Korea
2 Department of Radiology, Seoul National University Hospital, Seoul National University College of Medicine, Seoul, Republic of Korea
3 Department of Information Systems, Hanyang University, Seoul, Republic of Korea
4 Division of Pulmonary, Critical Care, and Sleep Medicine, Department of Internal Medicine, University of Kansas School of Medicine, Kansas City, KS, United States of America
5 Bioengineering Program, University of Kansas, Lawrence, KS, United States of America
6 Department of Medical Device Development, Seoul National University College of Medicine, Seoul, Republic of Korea
* Authors to whom any correspondence should be addressed.

E-mail: jchoi4@kumc.edu and changhyun.lee@snu.ac.kr

Keywords: graph neural network, graph classification, protein classification, matrix decomposition

Abstract

Recently, graph neural networks (GNNs) have been widely used in various domains, including social networks, recommender systems, protein classification, molecular property prediction, and genetic networks. In bioinformatics and chemical engineering, considerable research is being actively conducted to represent molecules or proteins on graphs by conceptualizing atoms or amino acids as nodes and the relationships between nodes as edges. The overall structures of proteins and their interconnections are crucial for predicting and classifying their properties. However, as GNNs stack more layers to create deeper networks, the embeddings between nodes may become excessively similar, causing an oversmoothing problem that reduces the performance for downstream tasks. To avoid this, GNNs typically use a limited number of layers, which leads to the problem of reflecting only the local structure and neighborhood information rather than the global structure of the graph. Therefore, we propose a structurally informed convolutional GNN (SICGNN) that utilizes information that can express the overall topological structure of a protein graph during GNN training and prediction. By explicitly including information of the entire graph topology, the proposed model can utilize both local neighborhood and global structural information. We applied the SICGNN to representative GNNs such as GraphSAGE, graph isomorphism network, and graph attention network, and confirmed performance improvements across various datasets. We also demonstrate the robustness of SICGNN using multiple stratified 10-fold cross-validations and various hyperparameter settings, and demonstrate that its accuracy is comparable or better than those of existing GNN models.

1. Introduction

Advances in machine learning (ML) and deep learning (DL) have made remarkable contributions not only to the fields of computer vision and natural language processing, but also to the development of bioinformatics and cheminformatics, such as molecular classification, molecular property prediction, and protein classification [1–3]. Adopting ML and DL methods to enhance the speed and accuracy of these classification and prediction tasks allows conserving both time and financial resources across various costly and time-consuming applications, including drug discovery; designing functional proteins such as enzymes, antibodies, transport proteins, and storage proteins; and exploring protein–protein interactions (PPIs) [4, 5]. Research on effectively representing molecules and proteins has been conducted across a wide range of disciplines and from diverse perspectives to improve the performance for these tasks. Traditional methods for classifying and predicting molecule and protein properties involve extracting fingerprints or hand-engineered features created by biological and chemical experts based on their domain knowledge and then applying regression or ML models. Although these methods have already demonstrated substantial performance gains and offer the advantage of utilizing representations based on well-studied biological and chemical theories, they rely heavily on domain expertise.

A popular method for expressing proteins and molecules is to represent each atom or amino acid as one character and then combining them to represent the proteins and molecules as a sequential string. For example, a molecule can be expressed using a simplified molecular-input line-entry system (SMILES) or self-referencing embedded strings (SELFIES), which are linear strings wherein the characters representing each atom are combined. Each protein can also be encoded as a string, which is a combination of characters, each corresponding to a distinct amino acid. Molecules and proteins expressed as sequential strings can be converted via various methods and used in downstream ML models for numerous biological and chemical tasks. For example, these sequential strings can be tokenized into individual characters or fragments and then used for property prediction and classification of proteins or molecules through sequence-processing models such as recurrent neural networks or long short-term memory (LSTM) [6–8].

In proteins and molecules, the structure between elements, such as amino-acid residues or atoms, is important and can be expressed as graph data, which can represent their topological structure better than sequential strings [9, 10]. In bioinformatics and cheminformatics, these graphs are used to represent various types of structures and interactions between entities. Residue graphs represent the amino-acid residues of a protein as nodes and the physical or chemical interactions between the nodes as edges [11, 12]. Similarly, molecular graphs represent the atoms of molecules as nodes and the bonds connecting the atoms are as edges [13–15].

Graphs can also be used to illustrate PPIs. Typically, verifying PPIs requires substantial financial resources and time. Nevertheless, the link-prediction task in the PPI network, wherein the connections expressing interactions between two proteins are predicted, enables significant time and financial cost savings by selecting potential proteins candidates that are likely to interact [16, 17]. Thus, by adopting graph representations, various graph-related ML and DL algorithms that are suitable for graph-format data can be employed to analyze molecular and protein data.

Innovative convolutional neural networks (CNNs), such as AlexNet [18], ResNet [19], and EfficientNet [20], have been exceptionally effective across diverse fields, including image classification, pattern recognition, and object detection. Recently, convolutional graph neural networks (GNNs), which learn node, edge, or graph representations through neural networks with convolution operations, have attracted considerable attention and are been actively researched. Image data can be regarded as special specific cases of graph data characterized by regular 2D grids, wherein each node represents a pixel and neighboring or adjacent pixels are connected to each other by edges. Similar to the convolution operation for one pixel of image data, which generates local features based on its neighboring pixel features, each node in a graph generates local features by receiving information from the surrounding neighboring nodes through convolutional GNNs. Through this process, the node or edge representation is learned by propagating or spreading information from neighboring nodes and aggregating the propagated feature values. Additionally, the entire graph representation can be learned by integrating the node and edge representations through an additional READOUT function. These learned representations are employed in various graph-related tasks, such as node classification, edge prediction, and graph classification. Depending on how information is propagated through neighboring nodes and how the propagated feature values are aggregated, diverse GNN models, such as graph convolutional network (GCN) [21], message-passing neural network (MPNN) [22], GraphSAGE [23], graph attention network (GAT) [24], and graph isomorphism network (GIN) [25], have been proposed.

In the GNN architecture, one layer propagates information through directly connected neighbors, implying the exchange of information with one-hop neighbors. By stacking k layers, each graph node can send or receive information from its neighbors up to k hops away. However, building a deeper GNN architecture by stacking more layers does not always guarantee better performance and may actually worsen it, as shown in figure 1. As a GNN model deepens, the embeddings of different nodes become more similar, which results in oversmoothing [21, 26, 27]. This issue is typically avoided by restricting the number of stacked layers when constructing a GNN model. As the structural characteristics of proteins are crucial for their classification, restricting the number of stacked GNN layers may lead to the omission of comprehensive global structural information.

In this study, we propose a structurally informed convolutional GNN (SICGNN) as a supplementary method that utilizes the global structural information of proteins when classifying them using a GNN. Given that proteins can be represented on graphs, which, in turn, can be depicted as an adjacency or Laplacian matrix, we exploited the structural information from protein graphs derived through matrix decomposition. We adopted the linear-transformation scale of the matrix, degree of transformation of the matrix

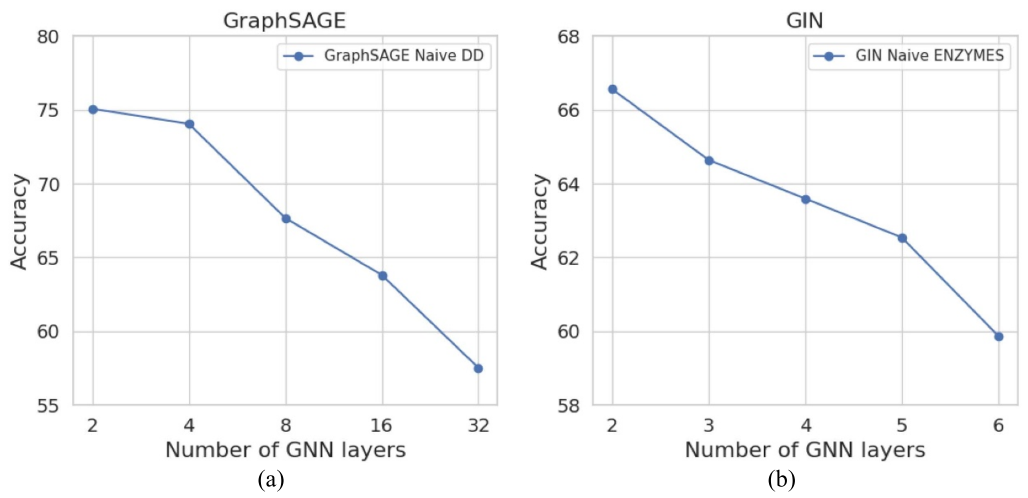


Figure 1. Oversmoothing occurs and classification accuracy decreases as the number of GNN layers increases in (a) GraphSAGE on DD dataset and (b) GIN on ENZYMS dataset.

eigenvectors, vector related to graph partitioning as structural information, and their compressed forms obtained through principal component analysis (PCA). We experimentally confirmed that the protein-graph classification accuracy of various GNN models can be improved by employing the global structural information of the graph for training each GNN and READOUT layer. Furthermore, we empirically validated the robustness of the proposed method through multiple stratified k-fold experiments.

The remainder of this paper is organized as follows. Section 2 provides an overview of protein-representation learning and GNN. Section 3 introduces the architecture of the proposed SICGNN model and illustrates its extensions and applications to representative GNN models. It also details the structural information of the graph used in our SICGNN model. Subsequently, section 4 presents the performance evaluation of the proposed SICGNN by comparing it with existing GNN models on benchmark protein-graph datasets. Finally, section 5 presents the conclusions and future research directions.

2. Related work

2.1. Protein-representation learning

In most fields, valuable representations or features from raw data are extracted manually by domain experts who identify important information and determine extraction methods based on their domain knowledge. By contrast, representation learning automatically extracts valuable characteristics and patterns from raw data. Recent developments in DL technologies have achieved innovative progress in representation learning, which has demonstrated outstanding performance in various computer vision tasks, such as image classification and object detection, and natural language processing tasks, such as translation and sentence generation. Therefore, DL is being actively employed for representation learning in the fields of bioinformatics and cheminformatics.

Proteins are crucial biomolecules that perform several key functions in living organisms such as enzymatic activity, transport, storage, signal transduction, immune responses, and membrane composition. The expression of these proteins and the derivation of useful information from these expressions for classification and function prediction are major hurdles in the field of bioinformatics.

There are two major protein-representation approaches. The first involves expressing them as amino-acid sequences, wherein amino acids are expressed as single characters and proteins are expressed as sequences of these characters. This approach is similar to the SMILES and SELFIES methods, which express each atom as a single character and a molecule as a sequential string of multiple atoms. The encoding of proteins represented as sequential strings of amino acids is similar to natural language processing text, and the simplest method is to convert the string from character-level to one-hot encoding [7, 28, 29]. Although this method offers easy implementation, it has limitations in expressing the similarity between the amino acids in the protein. A more sophisticated strategy is to learn vector representations that capture the similarities between proteins by applying the word-embedding algorithm Word2Vec [30] to amino-acid sequence strings [6, 7]. Recently, bidirectional contextual language models or LSTM-based structures have been used to learn protein-sequence embeddings [8, 31, 32]. Based on this representation method, various studies on predicting protein structures, functions, classifications, and interactions between protein pairs have achieved satisfactory results. Additionally, transformer models have recently shown good performance for protein-structure predictions [33, 34].

The second method involves depicting proteins on graphs, wherein the amino acids or residues constituting the proteins are expressed as nodes, and the interactions or relationships between amino acids or residues are expressed as edges [9, 35]. Thus, the edges signify the connectivity or relationships between amino acids or residues, and the protein is represented as an adjacency matrix. The chemical, structural, and functional properties of amino acids or residues, such as hydrophilicity, polarity, charge, and size, can be used as features of the amino-acid or residue nodes. Expressing a protein on a graph offers the advantage of reflecting its topological structure better than when it is expressed as a sequence. Because representation learning for sequentially expressed protein strings is achieved through word-embedding or natural language processing techniques, that for graph-based protein data can also be achieved via graph-DL methods. Various GNN models have been shown good performance on multiple protein benchmark datasets [21, 26, 27]. Graph structures are widely used in bioinformatics to express protein structures, which are compounds of amino acids and residues, and illustrate the relationships between proteins, such as PPIs or drug–protein interactions [17, 35, 36]. In these graphs, a node denotes a protein or drug and an edge indicates an interaction or association between a pair of proteins or a protein and a drug. They are mainly used to predict whether two proteins or a protein and drug will interact with each other, similar to a link-prediction task that predicts whether there is a relationship between two nodes.

2.2. GNNs

Breakthroughs in computer vision have been facilitated by the emergence of CNNs such as AlexNet [18], ResNet [19], and EfficientNet [20]. However, there are several problems in directly applying the convolution operation, which is the core operation of these models, to general graph-format data. First, although image and video data are generally less structured than tabular data, which are organized into rows and columns, graph data are even less structured. For example, when learning representations from image data, fixed-size information about the horizontal and vertical pixels of the image is available. However, in the case of graph data, each graph often has an arbitrary number of nodes and edges; therefore, graphs typically have different sizes. Second, the image has directions, such as up, down, left, and right, based on one pixel. In other words, there is only one pixel on the right-hand side or downward direction of a particular pixel. Therefore, when performing a convolution operation on an image, there is directionality in which the filter proceeds from left to right or top to bottom. However, the graph data do not include this directionality concept. To overcome these differences and apply convolution operations for graph-form data, two mainstream approaches have been developed: spectral and spatial.

Spectral convolutional GNN models define graph convolutions by introducing filters from the perspective of graph-signal processing [37, 38]. In the field of signal processing, the Fourier transform converts signals from the time domain to the frequency domain, which is possible because of the orthogonality of trigonometric functions. This allows analyzing the frequency composition of the signal. Similarly, given that the eigenvectors of the Laplacian matrices are orthonormal, they can be considered as Fourier basis and used for analyzing the composition of graph signals. From the perspective of graph-signal processing, a graph signal $x \in \mathbb{R}^n$ is defined as a feature vector of all nodes in a graph and x_i is the feature value of ith node. For undirected graphs, the adjacency matrix A represents the connections between nodes in a graph. Each position in the rows and columns represents a node in the graph. If a particular position in the matrix has a value of 1, it indicates that the node corresponding to that row is connected to the node corresponding to that column. If the value is 0, there is no direct connection between the nodes representing that row and column. Laplacian matrix $\mathbf{L}$ is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is a diagonal matrix of node degrees. As $\mathbf{L}$ is a real symmetric positive semi-definite matrix, it is decomposed into $\mathbf{L} = \boldsymbol{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathbf{T}}$, where $\boldsymbol{\Lambda}$ is a diagonal matrix of eigenvalues ($\boldsymbol{\Lambda} = \mathrm{diag}(\ [\lambda_0,\ \lambda_1 \ldots \lambda_{n-1}]\ )$) and $\mathbf{U}$ is an eigenvector matrix ordered by corresponding eigenvalues ($\mathbf{U} = [u_0, u_1, \ldots u_{n-1}] \in \mathbb{R}^{n*n}$). Here, the eigenvectors of $\mathbf{L}$ form an orthonormal space where $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$. In the signal-processing field, the convolution operation is typically defined by the inverse Fourier transform of the elementwise product of the Fourier-transformed values of signal $\mathbf{x}$ and filter $\mathbf{g}$. The spectral convolution operation within a graph follows a similar procedure and is described as follows:

$$\mathbf{x}{\star}_{\mathrm{G}}\mathbf{g} = \mathcal{F}^{-1}\left(\mathcal{F}\left(\mathbf{x}\right) \odot \mathcal{F}\left(\mathbf{g}\right)\right),$$

$$= \mathbf{U}\left(\mathbf{U}^{\mathbf{T}}\mathbf{x} \odot \mathbf{U}^{\mathbf{T}}\mathbf{g}\right),$$

where $\odot$ denotes the elementwise product. If we let a graph filter $\mathbf{g}_{\boldsymbol{\theta}} = \mathrm{diag}\left(\ \mathbf{U}^{\mathrm{T}}\mathbf{g}\ \right)$, the spectral graph convolution is expressed as

$$\mathbf{x}{*}_{\mathrm{G}}\mathbf{g}_{\theta} = \ \mathbf{U}\mathbf{g}_{\theta}\mathbf{U}^{\mathbf{T}}\mathbf{x}.$$

Various models such as spectral CNN [39], ChebNet [40], GCN [21], and CaleyNet [41] have been proposed based on the definition of $\mathbf{g}_\theta$. However, spectral-based methods have a critical drawback in that a new spectral analysis is necessary whenever the number of nodes in a graph changes. Thus, they are inappropriate for graph-classification tasks wherein the number and structure of nodes differ among graphs.

By contrast, spatial-based methods can be applied to graphs with different numbers of nodes; therefore, they can be used for both node- and graph-classification tasks. Spatial convolution operates on neighboring nodes, similar to the convolution operation of a CNN on images, wherein a filter is applied to a certain pixel and its neighbors to generate new feature values for that pixel. Spatial-convolutional GNNs operate in a similar manner by aggregating information from the central node and its neighbors to generate an updated representation of the central node. This can be regarded as information from neighboring nodes being propagated to the central node along the connected edges, or as information being passed from neighboring nodes to the central node. In other words, a node aggregates its feature and those received from its neighboring nodes and updates its own feature with a new value. Spatial-based convolutional GNN models comprise two important operations, AGGREGATE and UPDATE, and the models differ depending on the definitions of these operations. These models can be used for node-level tasks because the representation is learned for each node and can also be applied to graph-level tasks through the READOUT function, which extracts graph representations by integrating all node representations. Generally, the maximum, mean, and sum operations over all node representations are used as READOUT functions. Graph classification is possible through the READOUT operation in spatial-based convolutional GNN models such as GraphSAGE, GIN, and GAT [24, 25, 42].

In addition to the READOUT operation, entire graph embeddings can be obtained using a graph pooling operation, which reduces the graph size by sampling nodes to create a coarser graph. This is similar to image downsampling in a conventional CNN, wherein the pooling operation reduces the input size. Various pooling-based GNN models, such as DiffPool [43], SortPool [44], and SAGPool [45], have been proposed depending on how the number of nodes in the graph is reduced to create a coarser graph.

3. Proposed method

In this section, we propose a novel protein-graph classification method that utilizes the structural information of a graph for spatial-based convolutional GNN models without pooling. First, because our SICGNN model is modified through various GNN models, we outline its general architecture and compare it with that of a typical convolutional GNN. Next, we explain how graph-structure information is employed and integrated in GraphSAGE, GIN, and GAT, which are representative convolutional GNN models. Finally, we explain the utilized graph- structure information and its meaning within the context of our methodology.

3.1. Model architecture

A convolutional GNN with only one layer reflects only the information from one-hop neighboring nodes, whereas that comprising k layers reflects the information from all neighbors up to k-hop away. However, a GNN model with excessive layers suffers from oversmoothing, wherein the embeddings between all nodes become similar. Therefore, convolutional GNN models typically employ a limited number of layers to prevent oversmoothing. Although the global structure of a protein is a critical feature that may reflect its properties, using a limited number of layers in a GNN has a fatal disadvantage in that it contains only the local information and not the global structural information of the protein. To overcome this shortcoming, the proposed SICGNN model extracts global topological information from graph structures and employs it for learning and predicting spatial-based convolutional GNNs. The overall architecture of the proposed model is illustrated in figure 2. An input graph may consist of a single connected component or multiple disconnected components; a component is defined as a connected subgraph within a single graph. Regardless of whether it comprises one or several components, each graph represents a single biological unit. For instance, figures 2 and 3 show graphs with two components.

Each layer of a spatial-based convolutional GNN comprises two operations: the AGGREGATE operation, wherein each node receives information or messages from neighboring nodes and combines them into one, and the UPDATE operation, wherein the result of the AGGREGATE operation is combined with the feature of the current central node to calculate new feature values. These operations for the lth layer can be formulated as follows, which is similar to the MPNN framework [22, 46]:

$$m_{\mathcal{N}(u)} = \mathrm{AGGREGATE}^l\left(h_v^l | \forall v \in \mathcal{N}(u)\right), \tag{1}$$

$$h_u^{l+1} = \mathrm{UPDATE}^l\left(h_u^l,\ m_{\mathcal{N}(u)}^l\right). \tag{2}$$

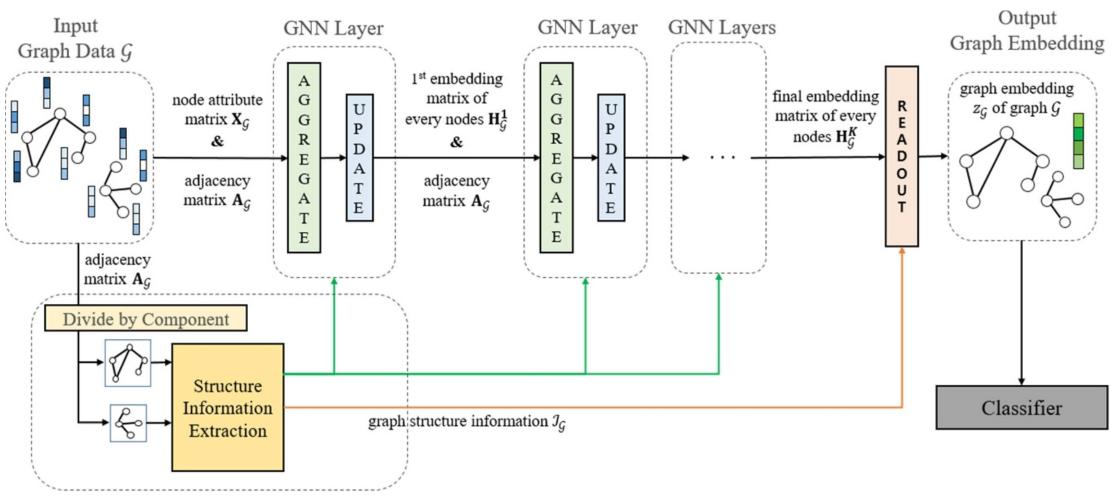

Input
Graph Data $\mathcal{G}$
node attribute matrix $\mathbf{X}_\mathcal{G}$ & adjacency matrix $\mathbf{A}_\mathcal{G}$
GNN Layer
AGGREGATE
UPDATE
1st embedding matrix of every nodes $\mathbf{H}^1_\mathcal{G}$ & adjacency matrix $\mathbf{A}_\mathcal{G}$
GNN Layer
GNN Layers
final embedding matrix of every nodes $\mathbf{H}^K_\mathcal{G}$
READOUT
Output
Graph Embedding
graph embedding $z_\mathcal{G}$ of graph $\mathcal{G}$
adjacency matrix $\mathbf{A}_\mathcal{G}$
Divide by Component
Structure Information Extraction
graph structure information $\mathcal{I}_\mathcal{G}$
Classifier

Figure 2. Overall architecture of the proposed SICGNN model. Each graph is divided into components and diverse structural information is extracted for each component. The extracted information is inserted into each GNN layer (green line) or READOUT layer (orange line) and used together for model training and prediction. Here, a graph with two disconnected components is given as input graph data.

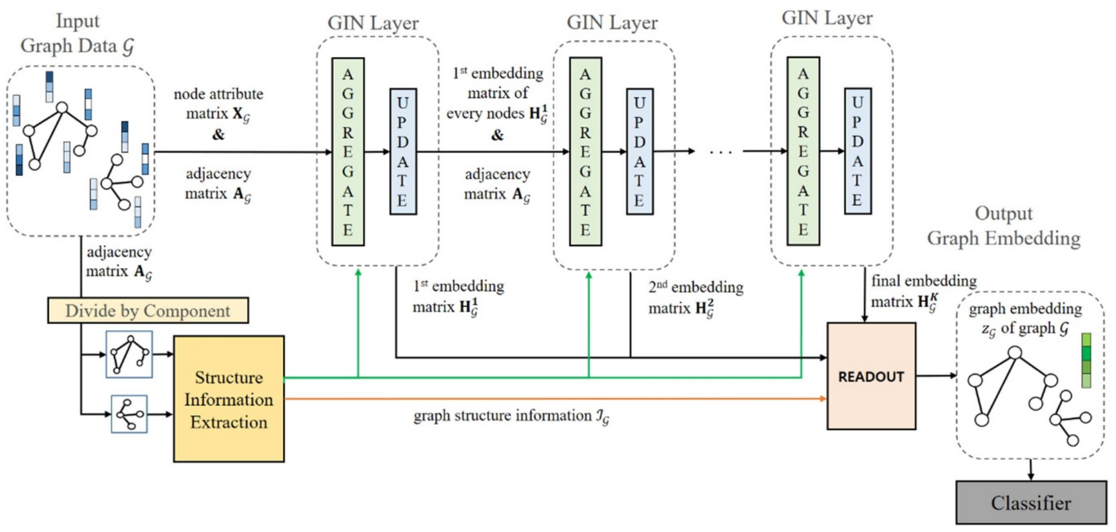

Input
Graph Data $\mathcal{G}$
node attribute matrix $\mathbf{X}_\mathcal{G}$ & adjacency matrix $\mathbf{A}_\mathcal{G}$
GIN Layer
AGGREGATE
UPDATE
1st embedding matrix of every nodes $\mathbf{H}^1_\mathcal{G}$ & adjacency matrix $\mathbf{A}_\mathcal{G}$
GIN Layer
GIN Layer
1st embedding matrix $\mathbf{H}^1_\mathcal{G}$
2nd embedding matrix $\mathbf{H}^2_\mathcal{G}$
final embedding matrix $\mathbf{H}^K_\mathcal{G}$
READOUT
Output
Graph Embedding
graph embedding $z_\mathcal{G}$ of graph $\mathcal{G}$
adjacency matrix $\mathbf{A}_\mathcal{G}$
Divide by Component
Structure Information Extraction
graph structure information $\mathcal{I}_\mathcal{G}$
Classifier

Figure 3. Overall architecture of the SIGIN model. Graph topological information is inserted into each GIN layer (green line) or READOUT layer (orange line). However, unlike the SIGraphSAGE shown in figure 2, which uses only the embedding of the last layer, SIGIN performs the READOUT operation for each GIN layer and then concatenates them.

For a GNN model comprising k layers, the embedding $z_\mathcal{G}$ of the entire graph $\mathcal{G}$ can be obtained by applying the READOUT operation to the kth embeddings of all nodes as follows:

$$z_\mathcal{G} = \mathrm{READOUT}(h^K_v \,|\, \forall v \in \mathcal{G}). \tag{3}$$

Our methodology involves extracting three types of structural information from the input protein-structure graph through several matrix decompositions. This information is utilized as additional structural information for each layer of the convolutional GNN or READOUT function and used together for learning and predicting graph embeddings. We propose using the global graph-structure information $\mathcal{I}_\mathcal{G}$ when each node performs the UPDATE operation with the message received through the AGGREGATE operation of equation (1) and its own embedding. This can be expressed abstractly at a high level by replacing equation (2) with the following:

$$h^{l+1}_u = \mathrm{UPDATE}^l\left(h^l_u, m^l_{\mathcal{N}(u)}, \mathcal{I}_\mathcal{G}\right). \tag{4}$$

Additionally, the READOUT operation can leverage the global graph-structure information, which can be expressed as follows by substituting equation (3):

$$z_{\mathcal{G}} = \mathrm{READOUT}(\, h_v^K, \mathcal{I}_{\mathcal{G}} \,|\forall v \in \mathcal{G}). \tag{5}$$

The proposed SICGNN model has the following three variants, depending on where the subsidiary graph-structure information is applied, and the best-performing model is considered as the final SICGNN model:

- SICGNN-A (applied to all layers): This model utilizes equations (4) and (5) instead of (2) and (3), respectively. This implies borrowing the graph-structure information for both the UPDATE and READOUT functions. Therefore, learning and prediction are performed by explicitly inserting the global structure information into each GNN layer, and the classifier can directly learn this information.
- SICGNN-R (applied to the READOUT layer): This model applies equations (2) and (5) instead of (4). Thus, the graph-structure information is utilized only in the READOUT layer and each layer of the model works in a manner similar to that of existing GNN models. The inclusion of graph-structure information in the final graph-embedding allows the classifier to explicitly learn the graph-structure information.
- SICGNN-I (applied to the intermediate layers): This model applies equations (3) and (4), but not (5). Therefore, the graph-structure information is explicitly used in the UPDATE operation. Thus, although each layer of the model explicitly incorporates the global topological structure information, the classifier used for graph classification does not directly leverage this information.

Algorithm 1 shows the pseudocode of the SICGNN with all the layers. Herein, both the green and orange lines in figure 2 are activated, and the topological information of the graph structure is reflected in each GNN and READOUT layer.

SICGNN-I and SICGNN-A explicitly incorporate global topological information into each layer, which can also be regarded as a form of skip connection, as proposed by ResNet [19] and SkipGNN [47]. Skip connection bypasses intermediate layers to deliver information directly, while still attempting to maintain the existing information. SICGNN-I and SICGNN-A endeavor to maintain global structural information by conveying it directly to each layer, thus preventing its attenuation. In addition, skip-connection uses a wider range of information at each layer by utilizing information not only from the layer immediately preceding it, but also from several preceding layers. SICGNN-I and SICGNN-A also employ information from the immediately preceding layer, as well as more expansive global information. This is analogous to skip connections in that it leverages diverse scope information. However, there are notable differences between the two approaches. In ResNet and SkipGNN, the skip connection strives to maintain the node embeddings between layers. In contrast, SICGNN-I and SICGNN-A aims to preserve the structural information about the graph.

3.2. Integrating convolutional GNN models with global structure information

In this subsection, we elucidate the method of applying global graph-structure information to convolutional GNNs such as GraphSAGE, GIN, and GAT, which are representative convolutional GNN models. Using the more abstract and general expressions (equations (4) and (5)), we describe the specific methods for each convolutional GNN model for implementation. To this end, we use the equations to express how each representative convolutional GNN model works, and how the global graph structure is combined.

3.2.1. GraphSAGE with graph-structural information

In GraphSAGE [23], which is an extended version of the GCN [21], each node samples and aggregates a fixed number of neighboring nodes rather than utilizing all neighboring nodes. However, unlike in social network graphs, the number of neighboring nodes of a specific node in protein graphs is not overwhelmingly large; therefore, information from all neighbors is generally employed rather than sampled for neighborhood aggregation [42]. We employed this approach to reach the entire neighborhood.

In the original GraphSAGE model, which employs the AGGREGATOR operation as the mean operator to unify the embeddings of neighboring nodes, the $l+1$th embedding of node u is expressed as follows:

$$m^l_{\mathcal{N}(u)} = \mathrm{mean}(\, h^l_v \,|\forall v \in \mathcal{N}(u)), \tag{6}$$

$$h^{l+1}_u = f\left(W^l \cdot \left[\, h^l_u || \, m^l_{\mathcal{N}(u)} \right] + b^l \right). \tag{7}$$

The aggregated message $m^l_{\mathcal{N}(u)}$ received by node u at the lth layer is the value obtained by applying the mean operator for the set of neighboring node embeddings, where the operator can be replaced with the max

Algorithm 1. SICGNN framework for generating graph embeddings.

```
Input: Graph $\mathcal{G}(\mathcal{V},\mathcal{E})$; node attributes $\mathbf{X}\{x_v, \forall v \in \mathcal{V}\}$; adjacency matrix $\mathbf{A}$; depth K, aggregator
       function $AGGREGATOR^l, \forall l \in \{1, 2, \ldots K\}$; update function $UPDATE^l, \forall l \in \{1, 2, \ldots K\}$
Output: Vector representation $z_{\mathcal{G}}$ of graph $\mathcal{G}$
1 structural information $\mathcal{I}_{\mathcal{G}} \leftarrow$ graph-structural information extracted from $\mathcal{G}$
2 $h_v^0 \leftarrow x_v, \forall v \in \mathcal{V}$
3 for $l = 0, 1, \ldots$ K-1 do
4     for $v \in \mathcal{V}$ do
5         $m^l_{\mathcal{N}(u)} \leftarrow AGGREGATE^l\ (\ h_v^l | \forall v \in \mathcal{N}(u)\ )$
6         $h_u^{l+1} \leftarrow\ UPDATE^l\left(\ h_u^l,\ m^l_{\mathcal{N}(u)},\ \mathcal{I}_{\mathcal{G}}\ \right)$
7     end
8 end
9 $z_{\mathcal{G}} \leftarrow READOUT(\ h_v^K,\ \mathcal{I}_{\mathcal{G}}\ |\forall v \in \mathcal{V}\ )$
10 return $z_{\mathcal{G}}$
```

or sum operator. The $l+1$th embedding of node u is updated through a linear combination and nonlinear activation function, such as the rectified linear unit, after concatenating the lth embedding of u and the message $m^l_{\mathcal{N}(u)}$ that it receives. Therefore, the UPDATE function for the general convolutional GNN in equation (2) is defined by equation (7) for GraphSAGE.

For a convolutional GNN comprising K layers, the embedding $Z_{\mathcal{G}}$ of graph $\mathcal{G}$ is obtained using the READOUT function after K iterations in the aforementioned process. Because GraphSAGE was not tested on graph-classification tasks in the original paper, we applied the max-pooling global READOUT function, as in previous studies [25, 42], to extract the embeddings of graph instances:

$$z_{\mathcal{G}} = \max - \text{pooling}\left(\left\{h_v^K | \forall v \in \mathcal{G}\right\}\right). \tag{8}$$

The UPDATE function of the GraphSAGE model was modified as follows to explicitly reflect the graph-structure information for each layer:

$$h_u^{l+1} = f\left(W^l \cdot \left[h_u^l \middle|\middle| m^l_{\mathcal{N}(u)} \middle|\middle| \mathcal{I}_{\mathcal{G}}\right] + b^l\right). \tag{9}$$

By applying topological information to GraphSAGE, we propose a structurally informed GraphSAGE (SIGraphSAGE) model with three variants: SIGraphSAGE-A, SIGraphSAGE-I, and SIGraphSAGE-R, depending on the layer in which the graph-structure information is exploited. SIGraphSAGE-A and SIGraphSAGE-I, which utilize the structural information in each GraphSAGE layer, use equation (9) to implement the UPDATE function, whereas SIGraphSAGE-R, which does not utilize the graph-structure information in each layer, uses equation (7). We also propose explicitly concatenating global structural information to the graph embeddings obtained through the READOUT operation of GraphSAGE as the final embeddings and inputting them into the classifier as follows:

$$z_{\mathcal{G}} = \left[\max - \text{pooling}\left(\left\{h_v^K | \forall v \in \mathcal{G}\right\}\right) | \mathcal{I}_{\mathcal{G}}\right]. \tag{10}$$

SIGraphSAGE-A and SIGraphSAGE-R are defined as the final embeddings with the results of the READOUT function and graph-structure information, as in equation (10). However, similar to the conventional GraphSAGE model, SIGraphSAGE-I only uses the results of the READOUT function for classification. This architecture is identical to that shown in figure 2.

3.2.2 GIN with graph-structural information

The GIN model performs graph classification based on the Weisfeiler–Lehman isomorphism test [48]. It uses the sum of neighboring node embeddings as the AGGREGATION function and a multilayer perceptron (MLP) network as the UPDATE function as follows:

$$m^l_{\mathcal{N}(u)} = \Sigma_{v \in \mathcal{N}(u)}\ h_v^l, \tag{11}$$

$$\begin{aligned} h_u^{l+1} &= \text{MLP}^l\left((1+\varepsilon) \cdot h_u^l + m^l_{\mathcal{N}(u)}\right) \\ &= \text{MLP}^l\left((1+\varepsilon) \cdot h_u^l + \Sigma_{v \in \mathcal{N}(u)} \cdot h_v^l\right). \end{aligned} \tag{12}$$

To obtain graph-level embeddings, GIN adopts a structure similar to a jumping knowledge network [49] and uses all the embeddings generated at all layers or iterations. Therefore, instead of using only the final-layer embedding, the READOUT function is applied to the embeddings of all layers as follows:

$$z_{\mathcal{G}} = \left[\, \text{CONCAT}\left(\text{READOUT}\left(\left\{ h_v^k | \forall v \in \mathcal{G} \right\} \right) | k = 0, 1, \ldots, K \right) \right] \tag{13}$$

After performing the READOUT function for each layer, they are concatenated and entered into the classifier as the final embeddings.

To adopt our methodology of utilizing the global graph-structure information at each layer of the GIN model, $\mathcal{I}_{\mathcal{G}}$ is added to the MLP input to generate h_u^{l+1} through the UPDATE operation as follows:

$$h_u^{l+1} = MLP^l \left(\left[\left((1+\varepsilon) \cdot h_u^l + \Sigma_{v \in \mathcal{N}(u)} \cdot h_v^l \right) || \mathcal{I}_{\mathcal{G}} \right] \right). \tag{14}$$

Thus, the graph-structure information can be explicitly reflected during learning and prediction in each GIN layer. Additionally, by integrating the graph-structure information with the final embedding results of the GIN, the following graph-level embedding, which also includes the structural information, is obtained:

$$z_{\mathcal{G}} = \left[\text{CONCAT} \left(\text{READOUT}(\{ h_v^k | \forall\, v \in \mathcal{G} \}) \, | k = 0,\ 1,\ \ldots K \right) \| \mathcal{I}_{\mathcal{G}} \right]. \tag{15}$$

The structure of the structurally informed GIN (SIGIN), which uses the embeddings from all layers, including the final layer, is shown in figure 3. Removing the green and orange arrows corresponds to the existing GIN model without structural information.

The SIGIN model also comprises three variants, SIGIN-A, SIGIN-I, and SIGIN-R, depending on where graph-structural information is additionally employed. Similar to the SIGraphSAGE variants, SIGIN-A employs equations (14) and (15), where both the UPDATE and READOUT operators additionally utilize the graph-structure information. By contrast, SIGIN-R uses equation (12), which represents the UPDATE operation of the original GIN, as the UPDATE operator, and equation (15), where only the READOUT operator is modified. By contrast, SIGIN-I uses equation (14) as the UPDATE operator, which reflects the structural information of each GIN layer, and equation (13), which represents the READOUT operation of the original GIN.

3.2.3. GAT with graph-structural information

GAT, which is the last model employed in our methodology, utilizes the attention score e_{ij} as defined in equation (16), which indicates the importance of node j to node i [24]. Here, j is calculated only for the neighbors of i. The normalized attention score α_{ij} is calculated by passing through the softmax function, as shown in equation (17), and a new embedding h_i^{l+1} of i is obtained as follows:

$$e_{ij}^l = \text{LeakyReLU}\left(a^T \left[W^l h_i^l || W^l h_j^l \right] \right), \tag{16}$$

$$\alpha_{ij}^l = \text{softmax}\left(e_{ij}^l \right) = \frac{\exp\left(e_{ij}^l \right)}{\Sigma_{k \in \mathcal{N}(u)} \exp\left(e_{ik}^l \right)}, \tag{17}$$

$$h_i^{l+1} = \sigma \left(\Sigma_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij}^l W^l h_j^l \right). \tag{18}$$

To obtain the entire embedding $z_{\mathcal{G}}$ of graph $\mathcal{G}$, we use max-pooling, similar to the GraphSAGE, as shown in equation (8). The architecture of the structurally informed GAT (SIGAT), which adds a global graph structure to the GAT model, is similar to the SIGraphSAGE architecture shown in figure 2. It comprises three variants, SIGAT-A, SIGAT-I, and SIGAT-R, depending on where it employs the structural information. To insert the graph-structure information into each layer of SIGAT-A and SIGAT-I, equations (16) and (18) are transformed into equations (19) and (20), respectively:

$$e_{ij}^l = \text{LeakyReLU}\left(a^T \left[W^l [h_i^l || \mathcal{I}_{\mathcal{G}}] || W^l \left[h_j^l || \mathcal{I}_{\mathcal{G}} \right] \right] \right), \tag{19}$$

$$h_i^{l+1} = \sigma \left(\Sigma_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij}^l W^l \left[h_j^l || \mathcal{I}_{\mathcal{G}} \right] \right). \tag{20}$$

To explicitly add global structure information to the graph-embedding obtained from the READOUT operation, we use equation (10), as in the case of SIGraphSAGE-A or SIGraphSAGE-R.

3.3. Explicit global graph-structure information

This subsection describes the global topological structure information explicitly added to each layer of the various GNNs or READOUT layers. Proteins can be represented as graph data comprising residues as nodes, and the graph data can be expressed as a matrix, such as an adjacency matrix. In this study, we extracted information from this matrix that accurately reflected the topological structure. The adjacency matrix $\mathbf{A}$ can be considered to reflect the properties of the graph because $\mathbf{A}f$, obtained by multiplying $\mathbf{A}$ by the graph node feature or graph signal f, indicates the propagation of the node feature or signal on the graph. Therefore, we propose using the eigenvalue, singular value, and Fiedler vector, which can be obtained through the decomposition of $\mathbf{A}$, as global structure information to be added to the representative convolutional GNN models.

We use eigenvalue λ as the first structural information of $\mathbf{A}$ that performs linear transformation, which satisfies the relationship ($\mathbf{A}v = \lambda v$) for eigenvector v. The geometric meaning of v is that it is a vector whose direction is preserved by linear transformation of $\mathbf{A}$. Thus, it is a scaling factor that indicates the extent to which $\mathbf{A}$ expands or contracts v. $\lambda > 1$ indicates that that the eigenvector is expanded, whereas $\lambda < 1$ indicates that it is contracted. Additionally, a negative λ value indicates that the vector direction is reversed. We use a list of eigenvalues $[\lambda_1, \lambda_2, \ldots \lambda_k]$ arranged in descending order together when training or predicting using convolutional GNNs, where k is a hyperparameter.

The second structural information, the singular values, is obtained through the singular value decomposition (SVD) of $\mathbf{A}$ and has been shown to be effective in image-data compression and graph-community detection [50–52]. SVD decomposes a graph matrix $\mathbf{A}$ of size $m \times n$ into $\mathbf{A} = \boldsymbol{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathbf{T}}$, and the singular values are diagonal elements of $\boldsymbol{\Sigma}$, which is a diagonal matrix of size $m \times n$. Here, $\mathbf{U}$ is an $m \times m$ orthogonal matrix that satisfies $\mathbf{A}\mathbf{A}^{\mathbf{T}} = \mathbf{U}\left(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{\mathbf{T}}\right)\mathbf{U}^{\mathbf{T}}$, and $\mathbf{V}$ is an $n \times n$ orthogonal matrix that satisfies $\mathbf{A}^{\mathbf{T}}\mathbf{A} = \mathbf{V}\left(\boldsymbol{\Sigma}^{\mathbf{T}}\boldsymbol{\Sigma}\right)\mathbf{V}^{\mathbf{T}}$. In other words, $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices obtained by the eigenvalue decompositions of $\mathbf{A}\mathbf{A}^{\mathrm{T}}$ and $\mathbf{A}^{\mathbf{T}}\mathbf{A}$, respectively. The protein data considered in this studyare symmetric matrices because they are undirected graphs. Hence, $\mathbf{A}$ has a size of $n \times n$, and $\mathbf{U}, \mathbf{V}, \boldsymbol{\Sigma}$ are matrices of size $\mathrm{n} \times \mathrm{n}$. Given that both $\mathbf{U}$ and $\mathbf{V}^{\mathrm{T}}$ are orthogonal matrices, they represent rotational linear-transformation matrices that do not change in size, but only in direction. Additionally, $\boldsymbol{\Sigma}$ is a diagonal transformation matrix that does not change in direction but only stretches or shrinks. Therefore, the linear transformation of multiplying $\mathbf{A}$ has the geometric meaning of rotating by $\mathbf{V}^{\mathrm{T}}$, scale transforming by $\boldsymbol{\Sigma}$, and then rotating again by $\mathbf{U}$. The singular values of a matrix indicate the degree of scale transformation owing to the linear transformation represented by this matrix and can be considered a unique feature of each protein graph.

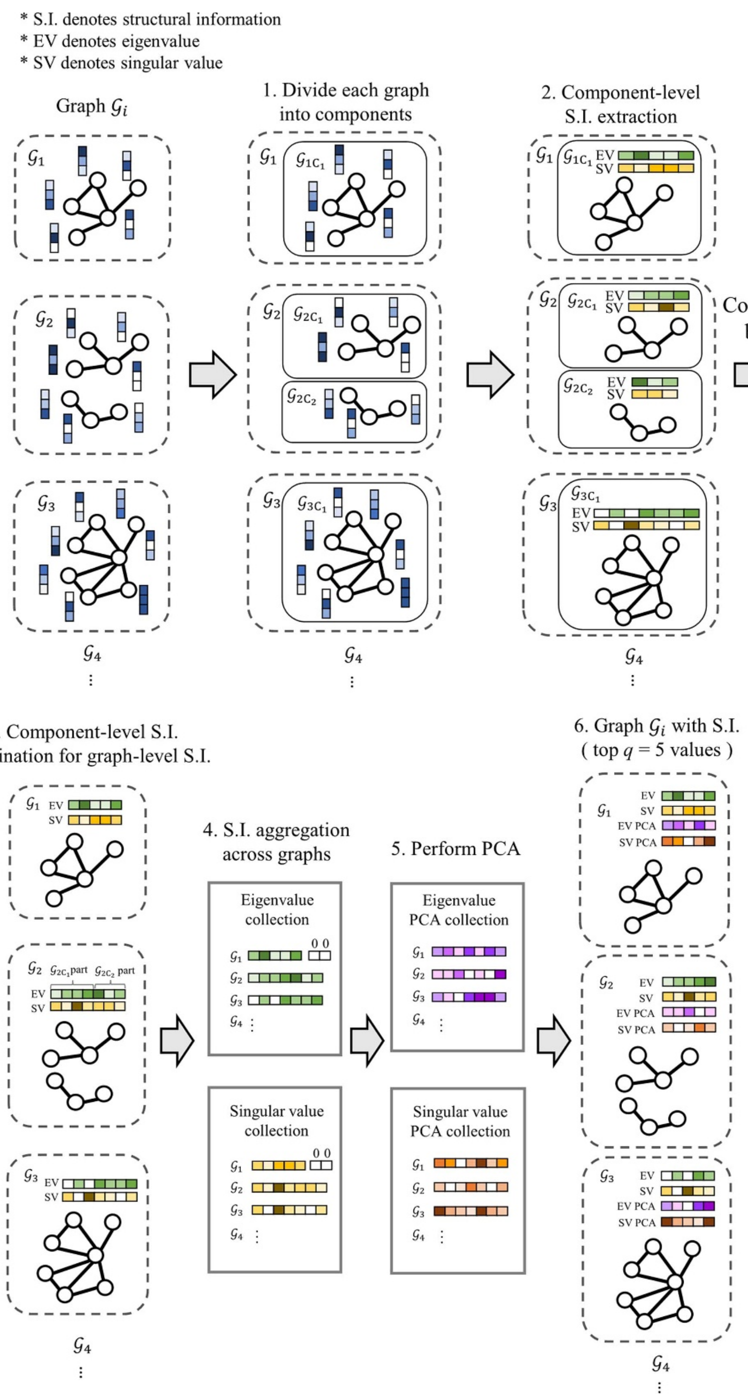
The third feature reflecting the topological structure of a graph is the Fiedler vector. It is an eigenvector corresponding to the nonzero smallest eigenvalue of the Laplacian matrix $\mathbf{L}$ defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is a diagonal matrix with the degree of each node of the graph as diagonal elements, and $\mathbf{A}$ is the adjacent matrix of the graph. Using the Fiedler vector, the nodes can be divided into two groups; thus, the graph can be partitioned into two subgraphs or clusters [53, 54].

The procedure for extracting the global structural information of the graph and performing PCA on the extracted data is illustrated in figure 4, with eigenvalues and singular values presented as in examples. Initially, each protein graph $\mathcal{G}_i$ is divided into components. Each graph $\mathcal{G}_i$ can comprise a single component, as illustrated by $\mathcal{G}_1$ and $\mathcal{G}_3$ in figure 4, or it can consist of two or more components that are disconnected from one another. As an illustration, $\mathcal{G}_2$ comprises two disconnected components. Consequently, the entire graph is subdivided into $\mathcal{G}_{2C_1}$ and $\mathcal{G}_{2c_2}$. In this case, the components are numbered in ascending order, starting with the component containing the largest number of nodes and concluding with the component containing the fewest number of nodes.

Secondly, at each component level, the eigenvalues and singular values are extracted which represent structural information of the component.

The eigenvalues (green) and singular values (yellow) derived here are independent of the node feature values (blue) of the original graph, and only reflect the topological structure of the graph. These nodal feature values play an important role in the node embedding process of the GNN; however, they are not used in the structural information extraction process.

Moreover, a component comprising n nodes will have an adjacency matrix of size $n \times n$, which will yield n eigenvalues and singular values for that component. For example, component $\mathcal{G}_{1C1}$ comprises five nodes, resulting in an adjacency matrix of size 5×5. From this matrix, five eigenvalues and singular values are extracted. In contrast, the component $\mathcal{G}_{2C1}$ contains four nodes and is represented by a 4×4 adjacency matrix, from which four eigenvalues and singular values are extracted. Accordingly, the number of nodes in each component determines the size of the adjacency matrix, which in turn determines the number of structural information.



* S.I. denotes structural information
* EV denotes eigenvalue
* SV denotes singular value
Graph $\mathcal{G}_i$
1. Divide each graph into components
2. Component-level S.I. extraction
Continued below
3. Component-level S.I. combination for graph-level S.I.
4. S.I. aggregation across graphs
5. Perform PCA
6. Graph $\mathcal{G}_i$ with S.I. (top $q = 5$ values)
Continued
Eigenvalue collection
Singular value collection
Eigenvalue PCA collection
Singular value PCA collection
EV
SV
EV PCA
SV PCA

Figure 4. Exemplary process of extracting the global structural information of the graph. Each of the graphs $\mathcal{G}_1$ and $\mathcal{G}_3$ is composed of a single component, whereas the graph $\mathcal{G}_2$ is composed of two disconnected components. The length of each structural information vector is set to $q = 5$. In step 3, to obtain the structural information of $\mathcal{G}_2$, the information of components with a large number of nodes comes first, and then the information of the smaller components is concatenated. In step 4, if the length of the structural information of graph is less than q, zero padding is incorporated to achieve the specified length.

Subsequently, the structure information of the individual components is aggregated into the overall graph information. In the case of $\mathcal{G}_1$, which is a graph comprising a single component, the graph structure information $\mathcal{I}_{\mathcal{G}_1}$ is identical to the component structure information $\mathcal{I}_{\mathcal{G}_{1C_1}}$. This is also true of $\mathcal{G}_3$. For graphs consisting of multiple disconnected components, the structural information of components with fewer nodes is concatenated behind the structural information of components with more nodes to represent a single graph structure. This means that the information from larger components with a greater number of nodes is given precedence. For instance, in graph $\mathcal{G}_2$, the eigenvalues and singular values of $\mathcal{G}_{2C2}$ is appended after that of $\mathcal{G}_{2C1}$ in order to express the structural information of graph $\mathcal{G}_2$.

In the next step, information from all graphs is collected for each structural information to form a singular value collection matrix and an eigenvalue collection matrix. In particular, if the total number of

Table 1. Statistics of benchmark protein-graph-classification datasets.

Dataset	# Graphs	# Classes	# Nodes	# Edges	# Max nodes
DD	1,178	2	284.32	715.66	5,748
ENZYMES	600	6	32.63	64.14	126
PROTEINS	1,113	2	39.06	72.82	620

graphs in the entire graph dataset is denoted by r, and the graph with the largest number of nodes has s nodes, then the collection matrices will have dimensions $r \times s$.

Given that the number of eigenvalues and singular values in each graph is contingent upon the number of nodes in that graph, we undertake a transformation of the structural information to align it with the node count of the graph with the largest number of nodes in the dataset. In order to achieve this, it is possible to insert graphs with an insufficient number of nodes into a matrix of consistent size by adding zero padding after them. For example, in figure 4, if we consider a dataset comprising 500 graphs, and the graph with the greatest number of nodes has 7 nodes, the resulting eigenvalue matrix or singular value matrix have dimensions of 500×7. In this instance, the structural information of both $\mathcal{G}_2$ and $\mathcal{G}_3$ can be placed in the matrix without alteration, as they already contain seven nodes. However, the structural information of $\mathcal{G}_1$, which contains five nodes, requires two additional zero paddings to achieve the same size before being inserted into the matrix.

The next step is to perform PCA on each of the generated eigenvalue and singular value collection matrices. It should be noted that the application of PCA may result in the appearance of non-zero values in the zero-padded areas. This results in a more comprehensive representation of each graph, as it encompasses both the original structural information, comprising eigenvalues and singular values, and the structural information compressed by PCA.

The length of this structural information is scaled to the number of nodes in the graph with the largest number of nodes in the dataset. However, utilizing all values is not optimal in terms of efficiency and performance. Accordingly, only the initial q values are employed for each structural information. Figure 4 depicts an example where $q = 5$ has been designated.

Additionally, when constructing a unified topological representation based on the topological information from graphs with multiple disconnected components, such as $\mathcal{G}_2$, it is possible to utilize a sorted version of the overall structural information. In this approach, each structural information value is ordered in descending sequence, regardless of which specific component it originates from. This approach allows for a more comprehensive examination, independent of the individual components to which these values may belong.

This approach is not based on the assumption that larger components exert a greater influence. Instead, it is a perspective that focuses on comparing the magnitude of the structural information values themselves. In other words, each individual structural information value is analyzed irrespective of the size of the component to which it belongs.

4. Results and discussion

4.1. Dataset and environment

We identified the three most commonly used benchmark protein-graph-classification datasets for evaluating the classification performance of GNNs: DD [55], ENZYMES [56], and PROTEINS [57]. Their statistics, such as the number of graphs, number of protein-graph classes, average number of nodes in each graph, average number of edges, and maximum number of nodes in a graph, are listed in table 1. DD is a collection of 1178 protein networks where the goal is to classify each protein class into enzyme and non-enzyme classes. Nodes in each graph represent amino acids.

PROTEINS consists of 1113 graphs, and each node is a secondary structure element (SSE) that represents the simple shapes that a protein chain forms due to interactions between surrounding atoms. This dataset is used to classify whether the protein graph is enzymes or non-enzymes. ENZYMES consists of 600 graphs, and each node represents a SSE. This dataset aims to classify each enzyme graph into one of 6 Enzyme Commission classes, where each class represents different types of enzyme activity. All datasets are available on [58].

All the GNN models were implemented in the Pytorch Geometrics library. To extract the graph topological structure information, the SciPy module was used for matrix decomposition and PCA was performed using the Scikit-learn module. Our experiments were conducted on a system comprising an Intel Xeon E5-2650 v4 CPU and an NVIDIA Titan XP 12 GB GPU.

Table 2. Performance comparison of representative GNNs and SICGNN models using stratified 10-fold cross-validation with multiple splits. Bold indicates the parts where either the naive model or our model achieves the better performance.

Model \ Dataset	DD		ENZYMES		PROTEINS	
	AVG Acc	SD Acc	AVG Acc	SD Acc	AVG Acc	SD Acc
GraphSAGE	75.021	3.424	58.142	7.215	73.876	3.546
SIGraphSAGE (ours)	**76.997**	**3.099**	**61.828**	**6.267**	**74.702**	**2.978**
GIN	75.257	2.679	63.594	4.853	73.702	2.698
SIGIN (ours)	**76.722**	**2.409**	**64.972**	**4.372**	**74.784**	**2.652**
GAT	72.767	2.854	52.589	**5.486**	73.989	**2.328**
SIGAT (ours)	**76.853**	**2.279**	**54.150**	6.331	**75.539**	2.823

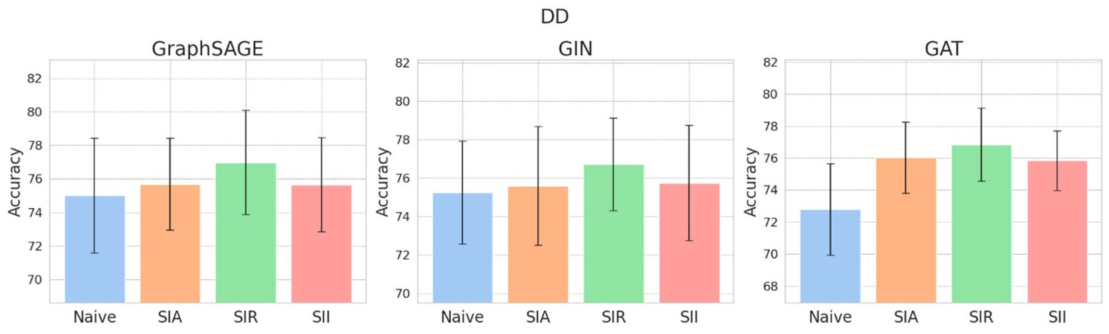
4.2. Performance comparison with original GNN models

For a more accurate and fair performance comparison, 10-fold cross-validation was conducted using the method proposed by Errica *et al* [42], and each dataset was divided into training, validation, and test sets in a ratio of 8:1:1. To demonstrate the robustness of our model, instead of performing 10-fold cross-validation only once, we conducted it three or more times and considered the average accuracy as the final measurement. To minimize the impact of an unfavorable random weight initialization, each test set was run more than five times, and the average was considered the final accuracy for that set. Additionally, all data splits were stratified to preserve the proportions of all classes across all 10-fold splits.

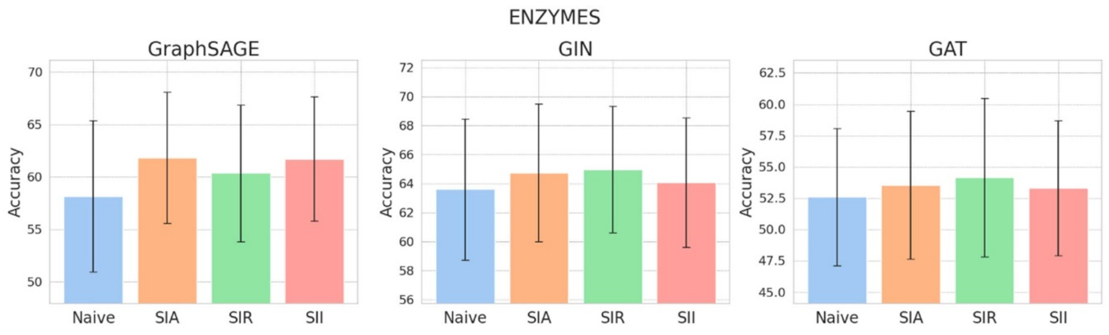
Table 2 presents a performance comparison of the representative GNN models introduced in section 2 and their SICGNN variants that use graph-structural information, which consistently achieved higher average classification accuracies. Additionally, they exhibited lower standard deviations, except for the SIGAT for ENZYMES and PROTEINS datasets. In particular, SIGraphSAGE and SIGIN showed consistently better predictive performance than naïve GraphSAGE and GIN, respectively, with higher average classification accuracies and lower standard deviations across all datasets. Although SIGAT exhibits slightly larger standard deviations for the ENZYMES and PROTEINS datasets than GAT, a certain performance improvement is evident owing to its higher average classification accuracies.

Figure 5 shows the performance comparisons among the three variants, -A, -I, and -R, of each SICGNN model of the GraphSAGE, GIN, and GAT models. Naïve GraphSAGE refers to the original GraphSAGE model and not the SIGraphSAGE variant that we proposed. Additionally, a similar performance comparison was conducted between the GIN and GAT. It is noteworthy that even though the proposed SIGraphSAGE model was selected as the best predictive among SIGraphSAGE-A, SIGraphSAGE-I, and SIGraphSAGE-R, all three variants showed superior or at least comparable classification accuracies compared to the naïve GraphSAGE model on all datasets. Furthermore, our GraphSAGE variants obtained lower standard deviations than naïve GraphSAGE on all datasets, indicating that the graph-structure information enhanced the performance of the naïve GraphSAGE model. Similarly, the GIN models with structural information demonstrated higher classification accuracies than the naïve GIN across all datasets. The standard deviations were also similar to or less than that of the naïve GIN. Finally, all three SIGAT variants performed better than the naïve GAT on the DD and ENZYMES datasets. However, for the PROTEINS dataset, only the SIGAT-R variant exhibited a performance improvement. This implies that in the case of the PROTEINS dataset, utilizing additional graph structure information in each layer of the GAT model may actually be counterproductive. It should be noted that the -R variant generally performed the best among the three variants.

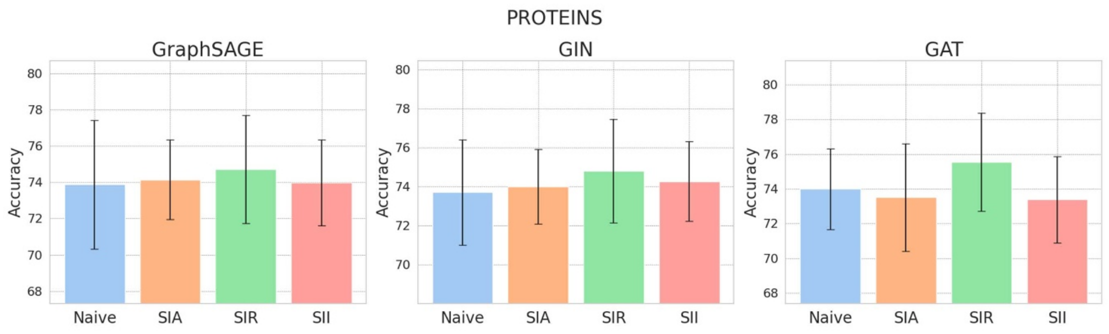
A detailed performance comparison of the SICGNN-A, -I, and -R models according to the type of graph-structure information for each dataset is presented in appendix A. Even for the same dataset, the graph-structure information that provides the greatest performance boost may differ among the base GNN models. For example, SIGraphSAGE-R, SIGIN-R, and SIGAT-R belong to the SICGNN-R model family, wherein only supplementary graph-structure information is applied in the READOUT layer. However, in the experiments on the DD dataset, the structural information that provided the largest performance gain differed among the model. For SIGraphSAGE-R, SIGIN-R, and SIGAT-R, beneficial structural insights came from the PCA of eigenvalues, PCA of singular values, and sorted singular values, respectively. In the case of the ENZYMES dataset, the PCA of singular values was the most beneficial for SIGraphSAGE-R and SIGAT-R, whereas the sorted singular values were the beneficial for SIGIN-R. For the same dataset, among the SICGNN-A variants, SIGraphSAGE-A showed the greatest performance improvement for the Fiedler vector, whereas SIGIN-A and SIGAT-A showed the highest performance gains for sorted singular values and sorted eigenvalues, respectively.



(a)



(b)



(c)

Figure 5. Classification accuracies and standard deviations of the naïve GNN model and its variants (SICGNN-A, SICGNN-I, and SICGNN-R) for the (a) DD, (b) ENZYMES, and (c) PROTEINS datasets. SIA, SII, and SIR stand for structural information added to all, intermediate, and READOUT layer, respectively. For the DD dataset, the accuracy of SIA, SIR, and SII is improved compared to the naive model of all three baseline GNNs, and SIR shows the best performance among all three models. Even in the ENZYMES dataset, SI variants of all three baseline GNNs always show good accuracy compared to the naive model. SIA performs best on GraphSAGE, and SIR performs best on GIN and GAT. In the PROTEINS dataset, for GraphSAGE and GIN, SI variants show similar or better performance than the naive model, and among them, SIR shows the best performance. However, although SIR improves performance in GAT, performance slightly decreases in SIA and SII. This shows that SICGNN-R often has the highest performance improvement, and that the layers to which additional structural information is useful may vary depending on the dataset and base GNN model.

Additionally, even with an identical GNN base, the graph-structure information affecting the performance improvement of each GNN layer and READOUT layer may be different. This means that the best-performing structural information might differ among the SICGNN-A, SICGNN-I, and SICGNN-R variants of GNN. For example, on the DD dataset, SIGraphSAGE-R showed the greatest performance gain by using the PCA of eigenvalues, whereas the sorted singular values were the most effective for SIGraphSAGE-I. For the PROTEINS dataset, the SIGIN-R yielded the highest classification performance using the PCA of singular values, but for SIGIN-I, sorted singular values were the most effective. Thus, no single type of structural information consistently improved performance across all datasets or of every GNN base model; thus, the ideal structural information for improving performance varies for each case.

Lastly, we conduct independent sample t-tests on the results presented in figure 5 to ascertain whether statistically significant performance improvements are observed with the SI variants compared to the naive model. The relevant t-statistics and p-values are presented in appendix B. It is worth noting that the SIR variant demonstrates statistically significant performance gains across all base models and datasets. Furthermore, in the DD dataset, both the GraphSAGE and GAT models demonstrate notable improvements

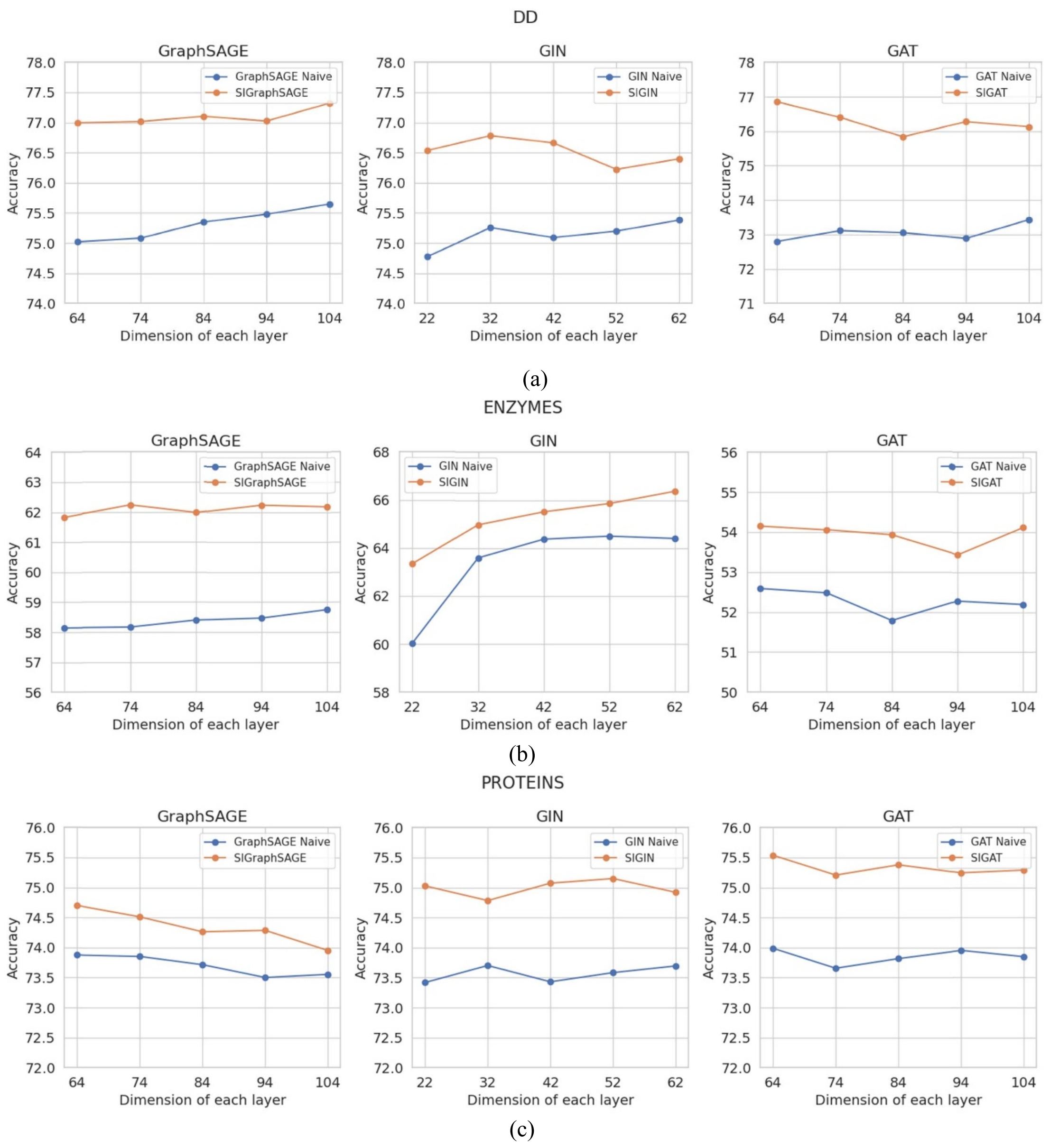


Figure 6. Performance comparison with changes in the dimensions of each GNN layer for the (a) DD, (b) ENZYMES, and (c) PROTEINS datasets. Despite model and dataset differences, the SICGNN model exhibits the higher classification accuracy than naïve GNN model. In the DD dataset (a), the accuracy of GraphSAGE slightly increases as the dimension increases. SIGraphSAGE roughly follows this trend but shows better performance. Although GIN and GAT do not have a clear trend according to dimension size, SIGIN and SIGAT outperform naive models. In the ENZYMES dataset (b), GraphSAGE and GIN show a slight increase in accuracy as the dimension increases. Likewise, SIGraphSAGE and SIGIN exhibit a similar trend, with higher accuracy overall. There is no clear trend in dimension size and accuracy for GAT and SIGAT, but SIGAT consistently outperforms naive GAT. In the PROTEINS dataset (c), the accuracy of GraphSAGE subtly decreases as the dimension increases, while SIGraphSAGE has a similar trend but shows higher accuracy. While GIN and GAT do not show clear trend between dimension size and accuracy, SIGIN and SIGAT consistently show higher accuracy.

in performance with the SIA, SII, and SIR variants. In the ENZYMES dataset, all three GNN models exhibit statistically significant improvements with the SIA and SIR variants.

4.3. Hyperparameter test

We compared the performances of the naïve GNN and SICGNN models by changing their hyperparameters to confirm the robustness of our methodology. Figure 6 shows the classification accuracy according to the GNN layer dimension size changes. Evidently, the proposed SICGNN approach consistently outperformed the naïve GNN model for every dataset, regardless of changes in the dimensions of the GNN layer.

Figure 7 compares the performances of the naïve GNN and SICGNN according to the changes in the number of layers of the GNN model. Although the SICGNN model tended to follow the performance of the

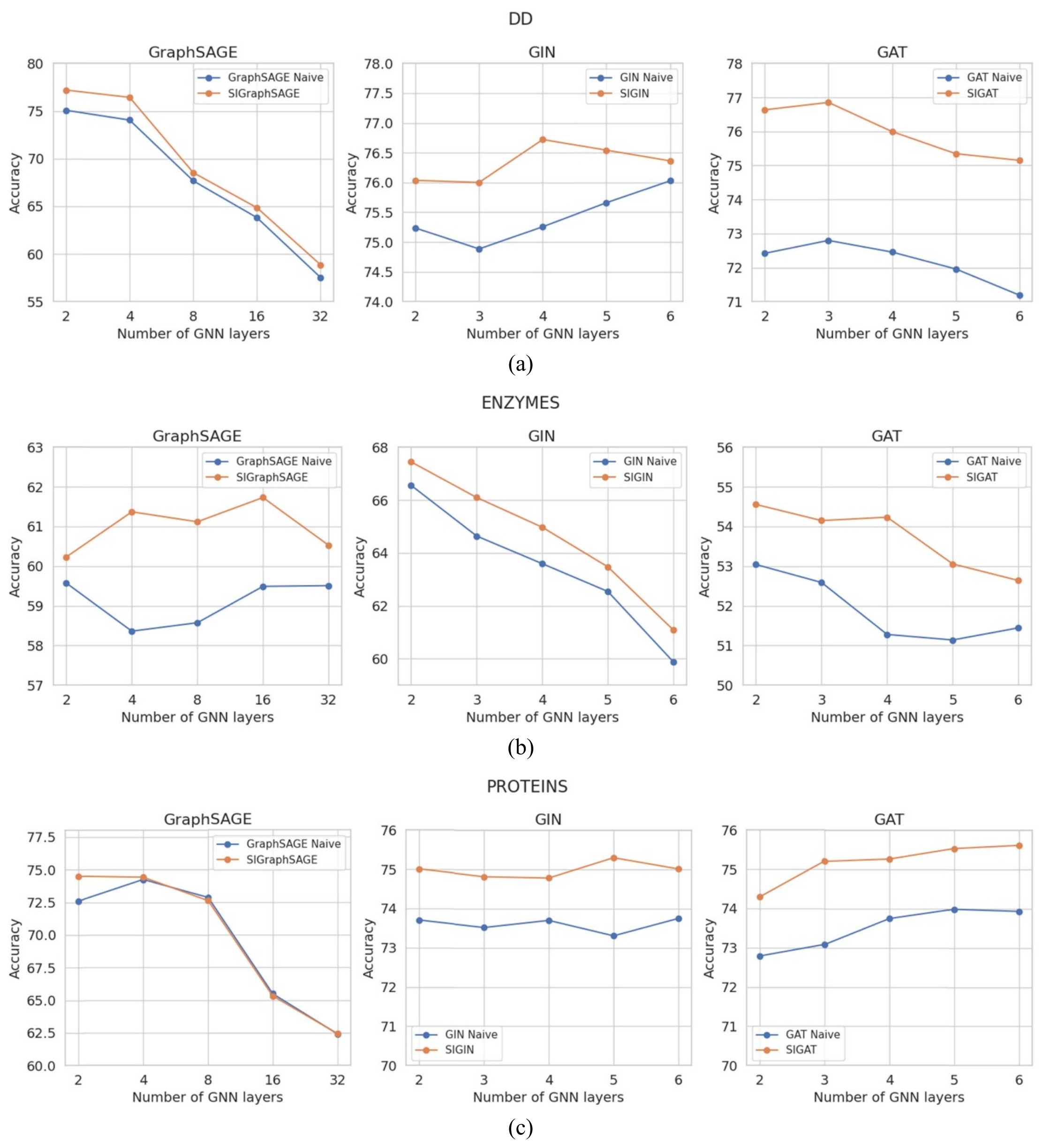


Figure 7. Performance comparison according to changes in the number of GNN layers. The performance changes in SICGNN tend to primarily reflect the performance changes in naïve GNN model, but it is usually superior or at least comparable. For DD dataset (a), GraphSAGE and GAT exhibit a decline in accuracy as the number of layers increases. Both SIGraphSAGE and SIGAT follow the same trend but with higher accuracy. In the ENZYMES dataset (b), increasing the number of layers results in reduced accuracy for GIN and GAT. SIGIN and SIGAT also follow this trend but consistently maintain higher accuracy. For PROTEINS dataset (c), accuracy in GraphSAGE decreases as the number of layers increases. SIGraphSAGE follows this trend, maintaining a comparable performance. In case of GIN, there is little change as the number of layer increases, though SIGIN achieves consistently better results. GAT shows gradual improvement with more layers, and SIGAT follows this upward trend with better performance.

underlying base-naïve GNN model, its accuracy generally tended to be superior or comparable. In these experiments, we observed that the proposed SICGNN model consistently outperformed the naïve GNN model for different hyperparameter values, thereby validating the robustness of our methodology.

5. Conclusion

We proposed a SICGNN architecture that utilizes the topological structural information of a graph derived from matrix decomposition to enhance the protein classification performance of convolutional GNN models. This methodology overcomes the fundamental problem of local access to the limited hops of neighbors in conventional architectures and facilitates the global structure of a graph. We propose three variations, -A, -I, and -R that indicate where the structural information is incorporated within the existing GNN architecture. The performance of our methodology was experimentally demonstrated to be better than or comparable to that of diverse representative GNN models, such as GraphSAGE, GIN, and GAT. The SICGNN consistently achieved comparable or higher accuracy than existing naïve GNNs, even under changes in the dimension size or the number of layers employed in the model.

We employed convolutional GNNs in which the graph structure was maintained. The other convolutional GNN (pooling-based GNN) induces graph coarsening, where the number of nodes in the graph gradually decreases, similar to the image-pooling operation in a CNN. We plan to utilize graph-structural information for these pooling-based GNNs in a future study. Additionally, we will attempt to fuse graph-structure information with graph transformer models.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/yhlee21/SICGNN.

Acknowledgments

This study was supported by the Mid-Career Bridging Program through Seoul National University, Grant 800-20220619, and was supported in part by the International collaborative research promotion program, SNU OPEN WORLD of Seoul National University.

Appendix A. Performance evaluation for each structural information.

Bold indicates the best performance among the structurally informed variants with the same base GNN and the same structure for the given dataset.

a. DD Dataset

		GraphSAGE		GIN		GAT	
	Model	Avg Acc	SD Acc	Avg Acc	SD Acc	Avg Acc	SD Acc
	Naïve	75.021	3.424	75.257	2.679	72.797	2.854
SIA	EV	75.219	3.135	75.268	3.393	73.941	2.522
	EVpca	74.706	3.268	72.764	3.123	**76.038**	**2.239**
	EVsorted	**75.690**	**2.748**	**75.604**	**3.082**	74.845	2.115
	Spectral	75.528	3.472	75.518	2.169	72.920	2.795
	Spectralsorted	75.064	3.712	75.460	3.323	72.850	3.475
	SV	75.653	2.990	75.541	2.848	74.777	2.417
	SVpca	74.791	2.794	73.787	2.946	75.473	3.079
	SVsorted	75.574	2.778	75.489	3.139	74.745	2.160
SII	EV	74.740	3.213	75.220	2.812	74.065	2.545
	EVpca	74.843	3.184	72.679	3.272	**75.860**	**1.869**
	EVsorted	75.428	2.715	**75.743**	**2.997**	74.388	2.330
	Spectral	75.070	3.287	75.710	2.281	72.378	2.645
	Spectralsorted	75.265	3.769	75.672	2.994	72.575	3.110
	SV	75.459	2.732	74.795	2.327	74.682	2.174
	SVpca	74.731	2.780	72.431	2.544	75.557	2.964
	SVsorted	**75.660**	**2.823**	75.484	3.685	74.570	2.289
SIR	EV	75.586	3.560	75.445	2.892	74.517	2.790
	EVpca	**76.997**	**3.099**	75.824	2.659	75.439	2.433
	EVsorted	75.739	3.506	75.846	2.733	76.776	2.343
	Spectral	75.732	3.541	75.315	2.223	72.810	3.125
	Spectralsorted	75.565	3.576	74.800	2.614	73.255	2.961
	SV	75.710	3.728	76.406	2.853	76.533	2.446
	SVpca	76.886	3.283	**76.722**	**2.409**	76.151	2.219
	SVsorted	75.720	3.448	75.712	3.096	**76.853**	**2.279**

b. ENZYMES dataset

		GraphSAGE		GIN		GAT	
	Model	Avg Acc	SD Acc	Avg Acc	SD Acc	Avg Acc	SD Acc
	Naïve	58.142	7.215	63.594	4.853	52.589	5.486
SIA	EV	59.811	5.365	64.375	4.683	53.400	6.245
	EVpca	55.322	4.766	62.514	4.265	52.500	4.870
	EVsorted	59.483	5.954	63.806	5.269	**53.539**	**5.886**
	Spectral	**61.828**	**6.267**	64.361	5.448	52.704	5.660
	Spectralsorted	60.233	7.048	64.236	4.296	52.718	5.493
	SV	60.289	5.858	64.542	4.220	52.537	5.481
	SVpca	56.433	5.267	62.708	5.192	53.142	5.193
	SVsorted	60.317	5.670	**64.736**	**4.740**	52.617	5.409
SII	EV	59.389	4.975	63.648	4.030	52.889	5.497
	EVpca	55.544	4.658	61.375	3.865	52.444	5.001
	EVsorted	59.194	6.103	63.111	4.886	52.181	5.462
	Spectral	**61.706**	**5.934**	64.056	4.629	53.139	5.704
	Spectralsorted	60.661	6.839	63.500	4.680	52.556	6.165
	SV	60.544	5.860	63.972	4.739	51.903	5.765
	SVpca	56.483	5.527	63.319	5.002	**53.311**	**5.400**
	SVsorted	60.522	5.981	**64.069**	**4.470**	52.778	5.563
SIR	EV	59.378	7.042	64.850	4.363	53.033	5.886
	EVpca	60.012	6.639	63.911	4.824	53.889	5.515
	EVsorted	59.433	6.354	64.178	4.659	53.339	6.082
	Spectral	59.050	6.967	64.300	4.623	52.894	5.623
	Spectralsorted	58.856	6.739	64.461	4.678	52.634	5.356
	SV	59.561	6.953	64.644	4.581	53.796	5.496
	SVpca	**60.344**	**6.518**	64.189	5.044	**54.150**	**6.331**
	SVsorted	59.144	6.890	**64.972**	**4.372**	53.356	6.069

c. PROTEINS dataset

	Model	GraphSAGE		GIN		GAT	
		Avg Acc	SD Acc	Avg Acc	SD Acc	Avg Acc	SD Acc
	Naïve	73.876	3.546	73.702	2.698	73.989	2.328
SIA	EV	73.663	2.803	73.234	1.826	72.417	2.978
	EVpca	73.736	2.282	72.896	2.735	72.597	3.244
	EVsorted	73.528	3.087	73.894	2.868	72.827	3.336
	Spectral	72.915	2.443	73.533	2.509	73.502	2.109
	Spectralsorted	73.583	2.190	**73.989**	**1.917**	**73.527**	**3.103**
	SV	73.833	2.483	73.610	2.774	72.722	2.385
	SVpca	73.968	2.576	72.238	2.597	71.933	3.103
	SVsorted	**74.143**	**2.203**	73.317	2.152	72.724	2.087
SII	EV	73.437	2.601	72.865	2.151	72.517	2.933
	EVpca	73.350	2.586	72.813	2.666	72.687	3.605
	EVsorted	73.463	2.941	73.901	2.573	72.631	3.114
	Spectral	72.919	2.527	73.494	2.995	**73.378**	**2.482**
	Spectralsorted	73.675	2.152	74.020	2.413	72.776	3.164
	SV	73.903	1.982	73.812	2.450	72.503	2.400
	SVpca	73.829	2.702	72.050	3.149	72.143	3.152
	SVsorted	**73.984**	**2.359**	**74.261**	**2.046**	72.493	2.723
SIR	EV	74.542	3.093	73.754	2.912	74.014	2.943
	EVpca	73.915	2.555	73.764	2.594	73.735	3.193
	EVsorted	74.493	3.006	74.004	2.522	74.444	3.594
	Spectral	74.298	3.172	73.733	2.452	74.259	2.260
	Spectralsorted	74.286	3.162	73.889	2.486	74.430	2.482
	SV	**74.702**	**2.978**	73.487	3.229	**75.539**	**2.823**
	SVpca	74.593	2.773	**74.784**	**2.652**	74.077	3.178
	SVsorted	74.264	3.145	73.744	2.539	75.363	2.638

Appendix B: Hypothesis testing for results on figure 4.

p-values less than 0.05 are highlighted in bold.

a. DD Dataset

GNN model / SI variation	GraphSAGE		GIN		GAT	
	t-statistic	*p*-value	*t*-statistic	*p*-value	*t*-statistic	*p*-value
SIA	−2.5837	**0.0100**	−1.0808	0.2811	−12.3448	**0.0000**
SII	−2.4940	**0.0129**	−1.5464	0.1236	−12.9135	**0.0000**
SIR	−7.4110	**0.0000**	−5.8548	**0.0000**	−15.2830	**0.0000**

b. ENZYMES dataset

GNN model / SI variation	GraphSAGE		GIN		GAT	
	t-statistic	*p*-value	*t*-statistic	*p*-value	*t*-statistic	*p*-value
SIA	−6.4785	**0.0000**	−2.2153	**0.0277**	−2.0450	**0.0413**
SII	−6.3993	**0.0000**	−0.9596	0.3382	−1.3299	0.1845
SIR	−3.8078	**0.0002**	−3.6540	**0.0003**	−3.2275	**0.0013**

c. PROTEINS dataset

GNN model / SI variation	GraphSAGE		GIN		GAT	
	t-statistic	*p*-value	*t*-statistic	*p*-value	*t*-statistic	*p*-value
SIA	−1.1078	0.2685	−1.225	0.2215	1.4587	0.1464
SII	−0.4392	0.6607	−2.2985	**0.0223**	3.0321	**0.0025**
SIR	−3.0896	**0.0021**	−4.9537	**0.0000**	−6.9605	**0.0000**

ORCID iDs

YongHyun Lee https://orcid.org/0000-0001-9887-9671
Eunchan Kim https://orcid.org/0000-0002-3743-3550
Jiwoong Choi https://orcid.org/0000-0002-7427-2979

References

[1] Li Z, Jiang M, Wang S and Zhang S 2022 Deep learning methods for molecular representation and property prediction *Drug. Discovery Today* **27** 103373
[2] Carracedo C J, Romero M C and Pérez R 2021 A deep learning approach for molecular classification based on AFM images *Nanomater* **11** 1658
[3] Zhang D and Kabuka M R 2020 Protein family classification from scratch: a CNN based deep learning approach *IEEE/ACM Trans. Comput. Biol. Bioinform.* **18** 1996–2007
[4] Ding W, Nakai K and Gong H 2022 Protein design via deep learning *Brief Bioinform.* **23** bbac102
[5] Notin P, Rollins N, Gal Y, Sander C and Marks D 2024 Machine learning for functional protein design *Nat. Biotechnol.* **42** 216–28
[6] Asgari E and Mofrad M R K 2015 Continuous distributed representation of biological sequences for deep proteomics and genomics *PLoS One* **10** e0141287
[7] ElAbd H, Bromberg Y, Hoarfrost A, Lenz T, Franke A and Wendorff M 2020 Amino acid encoding for deep learning applications *BMC Bioinform.* **21** 235
[8] Alley E C, Khimulya G, Biswas S, AlQuraishi M and Church G M 2019 Unified rational protein engineering with sequence-based deep representation learning *Nat. Methods* **16** 1315–22
[9] Giuliani A, Krishnan A, Zbilut J P and Tomita M 2008 Proteins as networks: usefulness of graph theory in protein science *Curr. Protein Pept. Sci.* **9** 28–38
[10] David L, Thakkar A, Mercado R and Engkvist O 2020 Molecular representations in AI-driven drug discovery: a review and practical guide *J. Cheminformatics* **12** 56
[11] Fasoulis R, Paliouras G and Kavraki L E 2021 Graph representation learning for structural proteomics *Emerg. Top. Life Sci.* **5** 789–802
[12] Ovchinnikov S and Huang P-S 2021 Structure-based protein design with deep learning *Curr. Opin. Chem. Biol.* **65** 136–44
[13] Wieder O, Kohlbacher S, Kuenemann M, Garon A, Ducrot P, Seidel T and Langer T 2020 A compact review of molecular property prediction with graph neural networks *Drug. Discovery Today Technol.* **37** 1–12
[14] Yang K *et al* 2019 Analyzing learned molecular representations for property prediction *J. Chem. Inf. Model.* **59** 3370–88
[15] Wu Z, Ramsundar B, Feinberg E N, Gomes J, Geniesse C, Pappu A S, Leswing K and Pande V 2018 MoleculeNet: a benchmark for molecular machine learning *Chem. Sci.* **9** 513–30
[16] Nasiri E, Berahmand K, Rostami M and Dabiri M 2021 A novel link prediction algorithm for protein-protein interaction networks by attributed graph embedding *Comput. Biol. Med.* **137** 104772
[17] Zhou H, Wang W, Jin J, Zheng Z and Zhou B 2022 Graph neural network for protein–protein interaction prediction: a comparative study *Molecules* **27** 6135
[18] Krizhevsky A, Sutskever I and Hinton G E 2017 ImageNet classification with deep convolutional neural networks *Commun. ACM* **60** 84–90
[19] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition *2016 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* pp 770–8
[20] Tan M and Le Q 2019 EfficientNet: rethinking model scaling for convolutional neural networks *Proc. 36th Int. Conf. on Machine Learning (PMLR)* pp 6105–14
[21] Kipf T N and Welling M 2017 Semi-supervised classification with graph convolutional networks Int. Conf. on Learning Representations (ICLR) (https://doi.org/10.48550/arXiv.1609.02907)
[22] Gilmer J, Schoenholz S S, Riley P F, Vinyals O and Dahl G E 2017 Neural message passing for quantum chemistry *Proc. 34th Int. Conf. on Machine Learning (PMLR)* pp 1263–72
[23] Hamilton W, Ying Z and Leskovec J 2017 Inductive representation learning on large graphs *Neural Information Processing Systems (NIPS)* vol 30
[24] Veličković P, Cucurull G, Casanova A, Romero A, Liò P and Bengio Y 2018 Graph attention networks *Int. Conf. on Learning Representations (ICLR)* (https://doi.org/10.48550/arXiv.1710.10903)
[25] Xu K, Hu W, Leskovec J and Jegelka S 2019 How powerful are graph neural networks? *Int. Conf. on Learning Representations (ICLR)* (https://doi.org/10.48550/arXiv.1810.00826)
[26] Chen D, Lin Y, Li W, Li P, Zhou J and Sun X 2020 Measuring and relieving the over-smoothing problem for graph neural networks from the topological view *Proc. AAAI Conf. Artif. Intell.* **34** 3438–45
[27] Liu M, Gao H and Ji S 2020 Towards deeper graph neural networks *Proc. 26th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining* (*New York, NY, USA*) (Association for Computing Machinery) pp 338–48
[28] Lin K, May A C W and Taylor W R 2002 Amino acid encoding schemes from protein structure alignments: multi-dimensional vectors to describe residue types *J. Theor. Biol.* **216** 361–5
[29] Kulmanov M, Khan M A and Hoehndorf R 2018 DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier *Bioinformatics* **34** 660–8
[30] Mikolov T, Chen K, Corrado G and Dean J 2013 Efficient estimation of word representations in vector space *Int. Conf. on Learning Representations (ICLR)* (https://doi.org/10.48550/arXiv.1301.3781)
[31] Rives A *et al* 2021 Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences *Proc. Natl Acad. Sci.* **118** e2016239118
[32] Elhaj-Abdou M E M, El-Dib H, El-Helw A and El-Habrouk M 2021 Deep_CNN_LSTM_GO: protein function prediction from amino-acid sequences *Comput. Biol. Chem.* **95** 107584
[33] Jumper J *et al* 2021 Highly accurate protein structure prediction with AlphaFold *Nature* **596** 583–9
[34] Yang J, Anishchenko I, Park H, Peng Z, Ovchinnikov S and Baker D 2020 Improved protein structure prediction using predicted interresidue orientations *Proc. Natl Acad. Sci.* **117** 1496–503
[35] Li M M, Huang K and Zitnik M 2022 Graph representation learning in biomedicine and healthcare *Nat. Biomed. Eng.* **6** 1353–69

[36] Zhong W, He C, Xiao C, Liu Y, Qin X and Yu Z 2022 Long-distance dependency combined multi-hop graph neural networks for protein–protein interactions prediction *BMC Bioinform.* **23** 521
[37] Shuman D I, Narang S K, Frossard P, Ortega A and Vandergheynst P 2013 The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains *IEEE Signal Process. Mag.* **30** 83–98
[38] Sandryhaila A and Moura J M F 2013 Discrete signal processing on graphs *IEEE Trans. Signal Process.* **61** 1644–56
[39] Bruna J, Zaremba W, Szlam A and LeCun Y 2014 Spectral networks and locally connected networks on graphs *Int. Conf. on Learning Representations (ICLR)* (https://doi.org/10.48550/arXiv.1312.6203)
[40] Defferrard M, Bresson X and Vandergheynst P 2016 Convolutional neural networks on graphs with fast localized spectral filtering *Neural Information Processing Systems (NIPS)* vol 29
[41] Levie R, Monti F, Bresson X and Bronstein M M 2019 CayleyNets: graph convolutional neural networks with complex rational spectral filters *IEEE Trans. Signal Process.* **67** 97–109
[42] Errica F, Podda M, Bacciu D and Micheli A 2020 A fair comparison of graph neural networks for graph classification *Int. Conf. on Learning Representations (ICLR)* (https://doi.org/10.48550/arXiv.1912.09893)
[43] Ying Z, You J, Morris C, Ren X, Hamilton W and Leskovec J 2018 Hierarchical graph representation learning with differentiable pooling *Neural Information Processing Systems (NIPS)* vol 31
[44] Zhang M, Cui Z, Neumann M and Chen Y 2018 An end-to-end deep learning architecture for graph classification *Proc. AAAI Conf. Artif. Intell.* **32** 1
[45] Lee J, Lee I and Kang J 2019 Self-attention graph pooling *Proc. 36th Int. Conf. on Machine Learning (PMLR)* 97 pp 3734–43 (available at: https://proceedings.mlr.press/v97/lee19c.html)
[46] Wu S, Sun F, Zhang W, Xie X and Cui B 2022 Graph neural networks in recommender systems: a survey *ACM Comput. Surv.* **55** 1–37
[47] Huang K, Xiao C, Glass L M, Zitnik M and Sun J 2020 SkipGNN: predicting molecular interactions with skip-graph networks *Sci. Rep.* **10** 21092
[48] Shervashidze N, Schweitzer P, Van Leeuwen E J, Mehlhorn K and Borgwardt K M 2011 Weisfeiler-lehman graph kernels *J. Mach. Learn. Res.* **12** 2539–61
[49] Xu K, Li C, Tian Y, Sonobe T, Kawarabayashi K and Jegelka S 2018 Representation learning on graphs with jumping knowledge networks *Proc. 35th Int. Conf. on Machine Learning (PMLR)* 80 pp 5453–62 (available at: https://proceedings.mlr.press/v80/xu18c.html)
[50] Kahu S and Rahate R 2013 Image compression using singular value decomposition *Int. J. Adv. Res. Technol.* **2** 244–8 (available at: https://www.academia.edu/4578584/Image_Compression_using_Singular_Value_Decomposition)
[51] Jaradat Y, Masoud M, Jannoud I, Manasrah A and Alia M 2021 A tutorial on singular value decomposition with applications on image compression and dimensionality reduction *2021 Int. Conf. on Information Technology (ICIT)* pp 769–72
[52] Sarkar S and Dong A 2011 Community detection in graphs using singular value decomposition *Phys. Rev.* E **83** 046114
[53] Shi J and Malik J 2000 Normalized cuts and image segmentation *IEEE Trans. Pattern Anal. Mach. Intell.* **22** 888–905
[54] Gatti A, Hu Z, Smidt T, Ng E G and Ghysels P 2022 Deep learning and spectral embedding for graph partitioning *Proc. 2022 SIAM Conf. on Parallel Processing for Scientific Computing (PP) (Proc.)* (Society for Industrial and Applied Mathematics) pp 25–36
[55] Dobson P D and Doig A J 2003 Distinguishing enzyme structures from non-enzymes without alignments *J. Mol. Biol.* **330** 771–83
[56] Schomburg I, Chang A, Ebeling C, Gremse M, Heldt C, Huhn G and Schomburg D 2004 BRENDA, the enzyme database: updates and major new developments *Nucleic Acids Res.* **32** D431–3
[57] Borgwardt K M, Ong C S, Schönauer S, Vishwanathan S V N, Smola A J and Kriegel H-P 2005 Protein function prediction via graph kernels *Bioinformatics* **21** 147–56
[58] Morris C, Kriege N M, Bause F, Kersting K, Mutzel P and Neumann M 2020 Tudataset: a collection of benchmark datasets for learning with graphs *Int. Conf. on Machine Learning (ICML)* (https://doi.org/10.48550/arXiv.2007.08663)