Medical Image Analysis (2024)

Contents lists available at ScienceDirect

Medical Image Analysis

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

Graph-in-Graph (GiG): Learning interpretable latent graphs in non-Euclidean domain for biological and healthcare applications

Kamilia Zaripova[a,*], Luca Cosmo[b,c], Anees Kazi[g], Seyed-Ahmad Ahmadi[d], Michael M. Bronstein[f], Nassir Navab[a,e]

[a]*Department of Computer Science, Technical University of Munich, Munich, Germany*
[b]*Department of Environmental Sciences, Informatics and Statistics, Ca' Foscari University of Venice, Venice, Italy*
[c]*Informatics Department, USI University of Lugano, Lugano, Switzerland*
[d]*NVIDIA, Munich, Germany*
[e]*Whiting School of Engineering, Johns Hopkins University, Baltimore, USA*
[f]*University of Oxford, Oxford, England*
[g]*Athinoula A. Martinos Center for Biomedical Imaging, Massachusetts General Hospital, Harvard Medical School, Boston, USA*

ARTICLE INFO

Article history:

2020 MSC: 68T07, 68T30

Keywords: Graph Deep Learning, Knowledge Discovery

ABSTRACT

Graphs are a powerful tool for representing and analyzing unstructured, non-Euclidean data ubiquitous in the healthcare domain. Two prominent examples are molecule property prediction and brain connectome analysis. Importantly, recent works have shown that considering relationships between input data samples has a positive regularizing effect on the downstream task in healthcare applications. These relationships are naturally modeled by a (possibly unknown) graph structure between input samples. In this work, we propose Graph-in-Graph (GiG), a neural network architecture for protein classification and brain imaging applications that exploits the graph representation of the input data samples and their latent relation. We assume an initially unknown latent-graph structure between graph-valued input data and propose to learn a parametric model for message passing within and across input graph samples, end-to-end along with the latent structure connecting the input graphs. Further, we introduce a Node Degree Distribution Loss (NDDL) that regularizes the predicted latent relationships structure. This regularization can significantly improve the downstream task. Moreover, the obtained latent graph can represent patient population models or networks of molecule clusters, providing a level of interpretability and knowledge discovery in the input domain, which is of particular value in healthcare.

1. Introduction

Recently, the number of applications with graph-based data have rapidly increased, in multiple domains such as computer vision (Zhou et al., 2020), computer graphics (Wang et al., 2019), physics (Shlomi et al., 2020), chemistry (Fung et al., 2021) and now healthcare (Dash et al., 2019). Here, Graph Neural Networks (GNNs) have proven to be a very powerful tool to process non-Euclidean unstructured data (Wu et al., 2019) in healthcare applications like disease prediction (Kazi et al., 2019a), drug interaction (Lin et al., 2020) and discovery (Li et al., 2017), brain connectome analysis (Kim et al., 2021) or multi-modal data analysis (Cosmo et al., 2020).

Most of the GNN models for graph classification consider each input graph individually. They aggregate information between neighboring nodes via message passing, to obtain new node or graph representations, which are then used for the final classification layers. On the other hand, recent literature on

*Corresponding author:
e-mail: kamilia.zaripova@tum.de (Kamilia Zaripova)

disease prediction attests that considering information coming from similar patients, encoded in the form of a population-level graph, can improve the overall model performance (Kazi et al., 2017). This is the case, for instance, of the method proposed in (Parisot et al., 2018) for brain connectome analysis, where single patient's brain graphs are converted into a vector representation and later leveraged in a population-level graph with each node as an individual patient associated to a feature vector representing the individual brain graph.

Despite showing the importance of considering the population level information to the downstream task, all these methods operate on a vectorial representation of the input data, which is obtained through an embedding of the input graph during preprocessing. As a consequence, the structural information from the individual graphs is lost because the pre-processing only focuses on the resulting feature vector. Moreover, the input samples in clinical or molecular prediction often incorporate crucial structural and functional information within the graph structure itself, which could be beneficial to the downstream task.

In this paper, we propose Graph-in-Graph (GiG), a GNN architecture that leverages both structural information and node-level features of the input graph samples, while learning a population-level graph between input samples. Importantly, the learned population-level graph provides a form of knowledge discovery, as it enables the inspection of the learned neighborhoods around input samples, and a level of reasoning about downstream decisions. Notably, most deep learning works optimize the downstream task only, and analyze the latent structure post-hoc through mapping techniques like t-SNE (van der Maaten and Hinton, 2008) or UMAP (McInnes et al., 2020). Instead, we aim to learn the downstream task and construct the latent graph structure end-to-end. To the best of our knowledge, combining latent graph learning with graph-valued input data has not been investigated so far. Furthermore, we propose a Node Degree Distribution Loss (NDDL) that regularizes the predicted latent relationship structure. This loss provides a parameter that allows the user to influence the connectivity degrees across samples. This can lead to better classification performance, but more importantly, it can facilitate knowledge discovery by forcing the model to maintain desired neighborhood sizes around input samples. We evaluate our method on real-world datasets from different domains, one medical and two bioinformatic, and provide a comparison with state-of-the-art graph classification models.

2. Related Work

Protein classification, brain connectome classification and toxicology analysis are among the most relevant biomedical applications in which the input data is naturally represented by a graph structure.

In the domain of protein classification, one of the first learning approaches was based on extracting feature vectors from proteins and classifying them into enzymes or non-enzymes using support vector machines (SVM) (Dobson and Doig, 2003). Later, variations were proposed, for instance using C-Support Vector Machines (Cortes and Vapnik, 1995) with graph kernels (Borgwardt et al., 2005). Recently, the use of Graph Convolutional Neural Networks (GCNNs) has been proposed as a general framework for graph and node representation learning, including proteins classification (Wang et al., 2019; Xu et al., 2019; Verma and Zhang, 2019). GNNs have been shown to capture local structures that are characteristic of the graph representation of the data (Nikolentzos and Vazirgiannis, 2020; Bouritsas et al., 2021; Cosmo et al., 2021; Bicciato et al., 2023). In particular, Zhang et al. (Zhang et al., 2019) reached SOTA results with the use of pooling operators and structure learning, which down-sample graph data and obtain a better embedding for the downstream classification tasks.

Brain connectome analysis requires pre-processing of brain signal time series from image/sensor data, in order to extract a connectivity graph between brain regions. Recent works in this field can be divided into three different parts: brain graph prediction (Sserwadda and Rekik, 2021; Zhang et al., 2020a,b; Bessadok et al., 2019b,a, 2021), brain graph integration (Gurbuz and Rekik, 2020; Demir et al., 2020) and brain graph classification (Song et al., 2020; Li et al., 2021; Xing et al., 2020; Wu et al., 2021; Chaari et al., 2022). For example, in the brain graph prediction domain, Mhiri et al. (Mhiri et al., 2021) proposed a method to predict a target brain graph from a source brain graph of different structure. They use the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) to minimize the gap between the two distributions and align the non-isomorphic source and target domains. MGN-Net (Gürbüz and Rekik, 2021) can be considered as a recent work on brain graph integration/fusion. It proposes a multi-view graph normalizer network (MGN-Net), which fuses multi-view graph population networks into a single connectivity template. Authors suggested to use the node strength distribution from the generated connectivity brain template, and compare it to the node strength distribution of randomly selected training samples via KL-divergence. Minimizing this distance reduces the deviation of the learned biological template and the randomized population networks. The most relevant works for us lie in the field of brain graph classification, which might be roughly split into single and multi-graph classification and multi-modal classification. For example, MICNet (Chaari et al., 2022) originally proposed to incorporate heterogeneous views to a subject-specific template graph. Chaari et al. constructed a meta-path adjacency matrix inspired by (Yun et al., 2019) and trained the model end-to-end for the subsequent classification task. Nebli et al. (Nebli et al., 2022) suggested reproducibility-based GNN selection (RG-Select) and evaluated recent multi-graph classification models for their capacity to reproduce the most discriminative features. DMBN (Zhang et al., 2020b) focused on multimodal data and performed sex classification using cross-modality learning. This was achieved by combining functional and structural brain networks via an encoding-decoding pipeline. In the domain of resting-state functional MRI (rs-fMRI), a method called rs-fMRI-GIN (Kim and Ye, 2020) was designed to make a classification on single-view graphs. Inspired by (Xu et al., 2019), a Graph Isomorphism Network (GIN) was adapted to rs-fMRI and functional connectivity analysis, as a dual representation of the convolutional

neural network. Pervaiz et al. (Pervaiz et al., 2020) suggested a unified brain data processing algorithm and evaluated it using different models such as BrainNetCNN (Kawahara et al., 2017) and ElasticNet (Zou and Hastie, 2005). ST-GCN (Gadgil et al., 2020) proposed to model the functional connectivity networks in rs-fMRI records with spatio-temporal graphs, by dividing rs-fMRI sequence into sub-sequences with a fixed window size. Most of these methods represent brain connectivity as a symmetric estimation of the functional relationships. In contrast, Mahmood et al. suggested creating a connectivity matrix at each time point via an attention mechanism and then aggregating crucial time points (Mahmood et al., 2022) (DECENNT).

Molecular toxicity prediction is a challenging task that often lies at the core of (de-novo) drug discovery. Consequently, it is well-studied, thanks also to open-access datasets like the Tox21 Data Challenge [1]. For example, DeepTox (Mayr et al., 2016) suggested using Deep Neural Networks (DNNs) and multi-task learning to predict molecular toxicity. In contrast, Dmlab (Barta, 2016) and Microsomes (Uesawa, 2016) provided tree-based ensemble methods as a solution. Class balance issues in the input data were addressed in (Idakwo et al., 2020), by suggesting different bagging approaches through resampling techniques. These models used input data in the Simplified Molecular-Input Line-Entry (SMILES) format. However, SMILES does not optimally preserve the molecular structure (Guo et al., 2021). Thus, bridging works were developed (Zaslavskiy et al., 2018; Guo et al., 2020) that combined different molecular fingerprints and molecular graphs as input. Recently, some models have been introduced using the molecular graph as input (Lu et al., 2019; Mansimov et al., 2019; Guo et al., 2021). For example, (Guo et al., 2021) suggested a meta-learning framework to obtain better results on a few samples. Proper pooling operations are another way to improve the graph representation and preserve the entire graph structure. For example, Graph Multiset Transformer (GMT) (Baek et al., 2021) used a multi-head attention mechanism to better preserve structural graph information.

A common factor in all these applications is the recent proliferation of methods applying GCNNs directly on the input graph representation. GCNNs have indeed shown to be a powerful tool to compute optimal node/graph representations. As such, they have become the preferred model for learning tasks in several domains, such as social sciences (Zhang and Chen, 2018; Qi et al., 2018), computer vision and graphics (Qi et al., 2017; Monti et al., 2017; Wang et al., 2019), physical (Choma et al., 2018; Duvenaud et al., 2015; Gilmer et al., 2017), as well as medical and biological sciences (Parisot et al., 2018, 2017; Mellema et al., 2019; Kazi et al., 2019c; Zitnik et al., 2018, 2019; Gainza et al., 2019). Another breakthrough in graph classification, especially in the domain of brain imaging and disease prediction, was the discovery that representations and downstream performances can benefit from considering relationships between patients, in the form of a population-level graph (Parisot et al., 2017, 2018; Vivar et al., 2018; Kazi et al., 2019a,b; Burwinkel et al., 2019; Vivar et al., 2021; Rakhimberdina et al., 2020). Parisot et al. (Parisot et al., 2017, 2018) first suggested to build a sparse population graph in which nodes represent imaging data and edges are weighted by a simple similarity metric on the patients' demographic or phenotypic information. Later, Kazi et al. (Kazi et al., 2019b) proposed constructing several graphs and parallel GCN branches, each corresponding to one of the demographic elements, and aggregating their node representations with a self-attention layer. Similarly, in (Rakhimberdina et al., 2020), it was proposed to combine separately constructed population graphs for diagnosing autism spectrum disorder. More recently, authors in (Jiang et al., 2019; Zhu et al., 2022) suggested learning the graph via a single-layer neural network parameterized by a weight vector. In (Cosmo et al., 2020), a latent graph learning module (LGL) was used to learn the population-level graph in form of a weighted adjacency matrix. In LGL, graph generation and the classification weights of the model are learned simultaneously and end-to-end. Finally, in (Kazi et al., 2020), LGL was expanded by training a probabilistic graph generator and sampler, which are able to produce sparse population graphs.

One of the limitations of the methods using population graphs is that they are mostly operating in a transductive paradigm and assume that the underlying graphs are known and fixed a priori or by construction (Parisot et al., 2017, 2018; Zhan et al., 2019; Li et al., 2018; Huang et al., 2018). This means that validation and/or test objects should be known during the training stage. In contrast, inductive methods assume that some nodes in the population graph might come in as new samples during test time (Cosmo et al., 2020; Kazi et al., 2020; Zhu et al., 2022). Our proposed GiG method is fully inductive and does not require test data during the training/validation stage. Another limitation derives from the fact that all the methods exploiting a population-level graph assume vector-valued input data of the input samples, either in the form of a flattened feature vector as input, or in the form of an embedding vector obtained from an input module. Importantly, this assumption is applied even when the input samples are naturally represented by a graph structure. On the other hand, SOTA methods for graph classification consider input graphs in isolation and none of the methods working in molecule classification consider a latent graph between molecules to obtain better representations.

We argue that preserving the initial graph structures at the input, and simultaneously learning their embedding in an end-to-end fashion, can be beneficial to obtain a better representation for downstream tasks and to extract only task-relevant information from feature vectors. Moreover, most datasets in the healthcare or bioinformatics domain suffer from an insufficient amount of data and missing information. In these cases, the learned latent graph can help recover missing information through neighborhood aggregation. This paper aims exactly at filling this gap, proposing a method that works directly on the graph representation of the input data and learns a population-level latent graph in an end-to-end fashion by optimizing the classification task loss.

[1] Tox21 Data Challenge URL: `https://tripod.nih.gov/tox21/challenge/about.jsp`

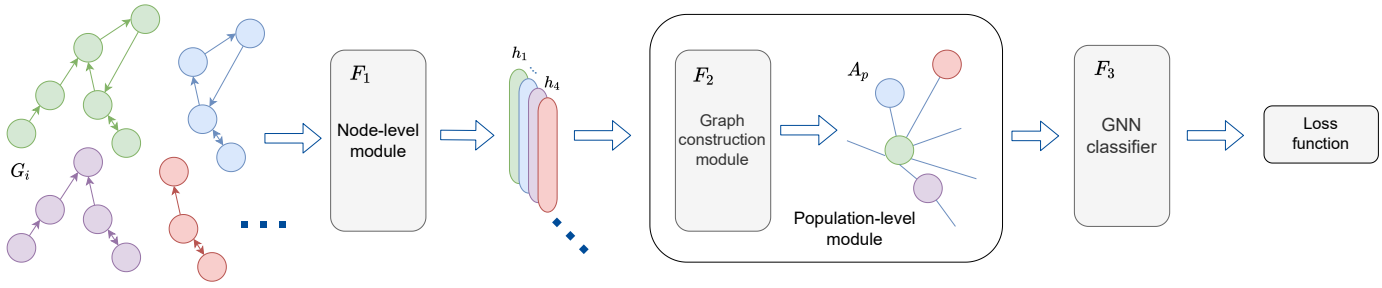


Fig. 1. The Graph-in-Graph architecture consists of three parts: F_1 represents the node-level module, F_2 the population-level module, and F_3 the GNN classifier. Graph-based inputs and their representations obtained from F_1 are indicated by $\mathbf{G_i}$ and $\mathbf{h_i}$ respectively, $i = 1..N$.

3. Method

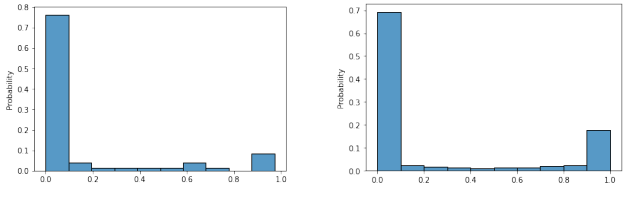
The input to our model is a set of N graphs, $\mathbf{G} = \mathbf{G_1}, \mathbf{G_2}, ..., \mathbf{G_N}$. Each i^{th} graph is defined as $\mathbf{G_i} = (\mathbf{V_i}, \mathbf{E_i}, \mathbf{X_i})$, where $\mathbf{V_i}$ and $\mathbf{E_i}$ are vertices and edges of the graph and $\mathbf{X_i} \in \mathbb{R}^{\|\mathbf{V_i}\| \times D}$ is the node feature matrix with D being number of features. The output of our model is a class probability vector $\mathbf{p} \in \mathbb{R}^C$ for each input graph, where C is the number of possible classes to which the input graph can be categorized.

3.1. Graph-in-Graph model

In this section, we provide the technical details of the proposed GiG model (Fig. 1). The proposed model consists of three parts: 1) the node-level module F_1; 2) the population-level module F_2; 3) the final GNN classifier F_3. In a nutshell, F_1 computes input graph representations, F_2 focuses on learning the latent connections between the input graphs, and F_3 takes as input both the graph representations from F_1 and the latent structure from F_2 and provides the predictions of the downstream task as output. In the following subsections, we provide technical details of each of the three parts.

3.1.1. Node-level module F_1

The node-level module is designed to directly handle non-Euclidian data (i.e. graphs) as input and captures only task-relevant features and structural information during training. Mathematically, F_1 is defined as $\mathbf{h_i} = F_1(\mathbf{G_i})$, where $\mathbf{h} = [\mathbf{h_1}, \ldots, \mathbf{h_N}]$ are output graph representations in an H-dimensional latent space, with $\mathbf{h_i} \in \mathbb{R}^{1 \times H}$. F_1 is implemented as a graph-convolutional neural network (GCNN) composed of a set of graph convolutional network (GCN) layers followed by a pooling operator over all node features. The specific choice of the graph convolutional operator to use in F_1 depends on the dataset and the task at hand (Section 4.2).



(a) LGL (b) LGL+NDD

Fig. 2. Distribution of values in the adjacency matrix $\mathbf{A_p}$ for PROTEINS$_{29}$ dataset with (b) and without (a) the degree distribution loss KL_{loss}. The x-axis indicates values of $\mathbf{A_p}$, the y-axis shows the values' occurrence probability.

3.1.2. Population-level module F_2

The population-level module is designed to learn the latent population graph. The output of F_1 i.e $\mathbf{h} = [\mathbf{h_0}, ...\mathbf{h_i}, ..., \mathbf{h_N}]$ is fed to F_2 as input. Each i^{th} node in the population level graph corresponds to the i^{th} input graph of the previous module, and it is represented by its representation $\mathbf{h_i}$. Formally, F_2 is a function defined as $\mathbf{A_p} = F_2(\mathbf{h})$, where $\mathbf{A_p} \in (0, 1)^{N \times N}$ is the weighted adjacency matrix encoding the population-level graph. Inspired by the population graph learning strategy of Latent Graph Learning (LGL) proposed in (Cosmo et al., 2020), we develop our LGL technique to let our model learn the population-level graph in an end-to-end fashion, together with minimizing the downstream task loss.

We learn the population-level graph by embedding each input graph representation $\mathbf{h_i}$ into a lower dimensional latent space in which the closeness of two node features indicates the existence of an edge between the two nodes in the population graph.

To project the input graphs to this latent space we define a learnable function g which takes the original graph representations $\mathbf{h_i}$ as input and yields a new representation $\tilde{\mathbf{h}}_\mathbf{i}$ belonging to this new latent space as output, $\tilde{\mathbf{h}}_\mathbf{i} = g(\mathbf{h_i})$. In our case, we use an MLP as our g function. The output of this module is then a weighted adjacency matrix $\mathbf{A_p}$ built from input graph representations $\tilde{\mathbf{h}}$. Precisely, the edge a_{ij} between the nodes i and j in $\mathbf{A_p}$ is defined as:

$$a_{ij} = \frac{1}{1 + e^{-t\left\|\tilde{\mathbf{h}}_\mathbf{i} - \tilde{\mathbf{h}}_\mathbf{j}\right\|_2 + \theta}}, \tag{1}$$

where θ and t are learnable soft-threshold and temperature parameters (Cosmo et al., 2020).

3.1.3. GNN classifier F_3

The output of the population-level module $\mathbf{A_p}$ is used by the final module (i.e. the GNN classifier), to produce the final prediction towards the downstream task. This later module is composed by the function $\mathbf{p} = F_3(\mathbf{h}, \mathbf{A_p})$, with $\mathbf{p} = [\mathbf{p_1}, \ldots, \mathbf{p_N}]$, which takes the individual graph representations $\mathbf{h}$ and the learned population-graph $\mathbf{A_p}$ as inputs and produces a class probability vector $\mathbf{p_i}$ for each of the input graphs as output.

Table 1. Dataset Statistics

Dataset	#Graphs	#Nodes	#Edges	#Node labels
HCP	1003	200	39800	200
$PROTEINS_{29}$	1113	39.06	72.82	29
TOX21	7831	18.6	38.6	9
D&D	1178	284.32	715.66	89
NCI1	4110	29.87	32.30	37
$PROTEINS_3$	1113	39.06	72.82	3
ENZYMES	600	32.63	64.14	3

Table 2. Class distribution for the datasets HCP, $PROTEINS_{3,29}$, D&D, and NCI1. The ENZYMES dataset samples are equally distributed among all 6 classes.

dataset	class0	class1	Graphs
HCP	469	534	1003
$PROTEINS_{3,29}$	663	450	1113
D&D	691	487	1178
NCI1	2053	2057	4110

Specifically, F_3 is composed of a set of graph convolution layers to produce the per-node embeddings, followed by two linear layers with shared weights and ReLU activation function in-between, to obtain the final output probabilities. F_3, being a graph convolution-based module, can leverage input sample similarities and obtain better representations for the downstream task.

3.2. Node Degree Distribution Loss (NDDL)

In this section, we propose a customized loss capable of achieving the downstream task and ensuring the quality of the population graph. We experimentally observed that optimizing the population-level module just using the task classification loss tends to produce graphs with adjacency matrix values close to either all-zero (highly disconnected) or to all-one (highly connected). An example of degree distributions in a graph with weak connections can be seen from the left bar plot in Fig. 2, i.e. an adjacency matrix $\mathbf{A_p}$ obtained after optimizing with the plain cross-entropy loss. As a consequence, the resulting graph is composed of many disconnected components and isolated nodes (Fig. 5, 6), which makes it difficult to reason about the predicted population graph structure.

To mitigate this problem, we propose to add a Node Degree Distribution Loss (NDDL) as a regularizer that helps produce more meaningful population graphs. The population-level adjacency matrix $\mathbf{A_p}$ is regularized by adding a prior to the desired node degrees distribution. The desired degree distribution is promoted by an additional Node Degree Distribution Loss (NDDL) based on Kullback–Leibler divergence between the computed degree distribution over $\mathbf{A_p}$ and the target distribution. In this work, we choose a Gaussian distribution as a target distribution, centered at the desired average node degree and defined in $R+$. This enforces a sparser graph while leaving the possibility of a few nodes having dense connections. We let the parameters (mean, standard deviation) of the Gaussian target distribution be optimized during the training to give the model more flexibility in adapting the population graph for optimal mean and standard deviation of node degrees.

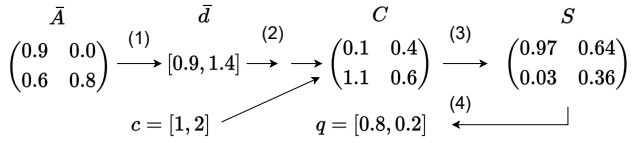


Fig. 3. The figure illustrates the transition from edge probabilities $\bar{A}$ to the node degree distribution q. The process consists of four steps: 1. Starting from the edge probability matrix $\bar{A}$ (2), we compute the soft node degree $\bar{d}$ along columns, i.e. as the sum of each row (3); 2. The distances $\Delta_{i,j}$ between node degree values c and $\bar{d}$ are computed and collected in the matrix C; 3. The soft assignment matrix S is computed by applying the softmax function on the squared values of C divided by σ^2 (4); 4. Finally, node degree distribution q is computed by normalizing the sum of the rows of matrix S (5).

On a symmetric adjacency matrix $\mathbf{A}$, the degree of the i^{th} node can be computed as the sum of the entries of the i^{th} row. Since $\mathbf{A_p}$ represents a weighted fully-connected graph, the actual degree (not considering self-loops) of each node is always $N-1$. To overcome this limitation, we exploit the fact that eq. (1) has the effect of pushing values in $\mathbf{A_p}$ either toward 0 or 1 and consider only the edges with a weight greater than 0.5 as actual edges of the graph:

$$\bar{\mathbf{A}} = \mathbf{A_p} * (\mathbf{A_p} > 0.5) \tag{2}$$

where $\bar{\mathbf{A}} \in \mathbb{R}^{N \times N}$ is the adjacency matrix used to compute the node degree of each node as:

$$\bar{d}_j = \sum_{i=1}^{N} \bar{\mathbf{A}}_{i,j} \tag{3}$$

with $\bar{d}_j \in \mathbb{R}^N$. Since $\bar{d}_j$ can assume a continuous value between 0 and $N-1$ (or between 1 and N, assuming self-loops), to compute the discrete degree distribution, we perform a soft assignment $S_{i,j}$ between d_j and the possible discrete node degree values $c_i, c = [1, \ldots, N]$, and compute the soft assignment matrix S values as:

$$S_{i,j} = \frac{e^{\frac{-\Delta_{i,j}^2}{\sigma^2}}}{\sum_k e^{\frac{\Delta_{k,j}^2}{\sigma^2}}} \tag{4}$$

where σ is a hyperparameter that we experimentally set to 0.6 and $\Delta_{i,j} = c_i - \bar{d}_j$. The value corresponding to the degree c_i in the computed node degree distribution $q = [q_1, ..., q_N]$ is:

$$q_i = \frac{\sum_j S_{i,j}}{\sum_{k,j} S_{k,j}} \tag{5}$$

Finally, the *NDDL* term is defined as

$$NDDL = D_{KL}(q, r) \tag{6}$$

where r is the target normal discrete distribution with learnable parameters. The NDD loss is then added to the Cross-Entropy

loss and used to train the model as a penalty term weighted by α:

$$loss = CE_{loss} + \alpha NDDL \tag{7}$$

In particular, considering the classification task, the Cross-Entropy loss is defined as

$$CE_{loss} = -\sum_{c=1}^{C} label_c * log(p_c) \tag{8}$$

where $label_c$ is the class membership indicator (i.e. has value 1 if the sample belongs to the class c, 0 otherwise) and p_c is the predicted class probability. For illustration purposes, the computation process of the node degree distribution can be seen in Figure 3.

4. Experiments and results

This section shows experiments on biological, medical, and chemical applications where each input sample is represented as a graph. In particular, we choose PROTEINS (Kersting et al., 2016), HCP (Essen et al., 2013) and Tox21 (Wu et al., 2018) datasets. Further, we make use of the publicly available benchmark datasets D&D (Dobson and Doig, 2003), NCI1 (Wale et al., 2008) and ENZYMES (Schomburg et al., 2004) to compare GiG with other general-purpose graph classification methods. These datasets are commonly used in the evaluation of graph classification models for binary and multi-class classification. In all the benchmark datasets, we follow the training/testing procedure described in (Errica et al., 2020) and use the same splits for a fair comparison.

We show two variants of the GiG model. In the first variant (LGL) the population-level graph is learned end-to-end using just the cross-entropy loss for the final classification task as in (Cosmo et al., 2020). In the second variant (LGL+NDD) the model incorporates the proposed Node Degree Distribution Loss (NDDL) (3.2) along with learning the population-level graph. In order to better validate the benefit of the population-level module, we also investigate different possibilities to build or learn the population graph.

4.1. Datasets

This section gives the details of the datasets used for our experimental evaluation. We used HCP, PROTEINS_{29} and Tox21 datasets to investigate the proposed framework and to evaluate the influence of each part of the GiG model and D&D (Dobson and Doig, 2003), PROTEINS_3 (Borgwardt et al., 2005), NCI1 (Wale et al., 2008) and ENZYMES (Schomburg et al., 2004) to compare our method with standard graph classification methods. Detailed statistical information on each dataset can be found in Tables 1, 2. For the precise description and preprocessing steps of the benchmark datasets (D&D, NCI1, PROTEINS_3, ENZYMES), please refer to (Errica et al., 2020).

HCP. The Human Connectome Project (HCP) is a medical dataset consisting of 1003 complete resting-state functional MRI (fMRI) runs released in 2017 (Essen et al., 2013). The task is to predict the sex of each patient based on the corresponding brain fMRI data. In (Pervaiz et al., 2020), authors analyzed several preprocessing steps for HCP datasets. Following their analysis, we adopt the Group Independent Component Analysis (GroupICA) parcellation scheme (Smith et al., 2014; Beckmann and Smith, 2004), and minimally pre-processed data, in gray ordinates with ICA FIX denoising (Salimi-Khorshidi et al., 2014). Detailed pre-processing steps can be found in (Pervaiz et al., 2020). A correlation matrix was computed using Pearson's correlation coefficients, which were calculated between fMRI signals from each of the 200 parcellations in the brain, to obtain functional connectivity on brain graphs. As a result, an affinity matrix of size 200 x 200 is associated with each brain.

PROTEINS. PROTEINS is a binary classification dataset (Borgwardt et al., 2005) provided by Technical University Dortmund (Kersting et al., 2016). The task is to classify proteins as enzymes or non-enzymes. We used two versions of the PROTEINS dataset: PROTEINS_{29} and PROTEINS_3, where the subscript indicates the number of features used in the feature matrix. For population graph analyses, we use the CATH Superfamily information (Orengo et al., 1997; Pearl et al., 2000) of each protein, which was retrieved from (Sillitoe et al., 2020). Importantly, CATH labels were not used during the training stage. We used CATH labels only to investigate and discuss potential knowledge discovery of population-level graphs. The CATH class hierarchy consists of several levels. We consider only the first class level (C-level) with three major classes: "mainly-alpha", "mainly-beta", and "alpha-beta".

Tox21. Tox21 is a public dataset (Wu et al., 2018) consisting of 8014 toxicity measurements on 7 nuclear receptor signals and 5 stress response panels. The task is to predict the binary value of each of the 12 targets that might be equal to "1" or "0", representing the "active" or "not active" state.

4.2. Implementation details

GiG models are trained in the usual supervised fashion. The population graph is constructed separately for every training/validation/test batch. The batch size at test and training time might be different and independent of each other. Our method is fully inductive, i.e. the test data is unknown during the training/validation steps. All the experiments are trained with the Adam optimizer (Kingma and Ba, 2015). For each dataset, the number of layers in each module, learning rate, batch size, alpha in (7) and the global pooling operator (mean or add) are tuned based on the validation loss (Table 10). Following (Errica et al., 2020), we optimized hyper-parameters for each fold of each benchmark dataset (Table 11). We used ReLU (Agarap, 2018) as the activation function for all the datasets, except KNN experiments on the HCP dataset, which were further optimized by adding batch normalization after the node-level module and after each layer in the GNN module, and by changing ReLU activation function to sigmoid, in order to obtain the best performing model. We used different learning rates for the optimizer, μ and σ, and θ and t (Tables 10, 11), as well as two types of learning rate schedulers depending on the dataset (Reduce Learning Rate on Plateau with threshold mode='abs', Cosine Annealing (Loshchilov and Hutter, 2017)). In Table 3, models have been trained using GraphConv (Morris et al., 2019) layers

Table 3. Comparison of dataset with several baselines and classical (cl.) methods. DECENNT (Mahmood et al., 2022), HGP-SL (Zhang et al., 2019), ToxicBlend (Zaslavskiy et al., 2018) classical methods for HCP, PROTEINS and Tox21 respectively. The top three performance is shown in boldface, blue, and violet.

	Without population graph		*Non-learned population graph*			*Learned population graph*	
	cl. methods	GCN	Random	KNN	GiG DGCNN	GiG LGL	GiG LGL+NDD
HCP (acc %)	86.0 ± NA	84.2 ± 2.5	43.7 ± 8.4	85.87 ± 2.6	89.0 ± 3.3	89.4 ± 1.3	**89.7 ± 1.3**
$PROTEINS_{29}$ (acc %)	**84.9 ± 1.6**	80.0 ± 2.5	53.7 ± 8.4	75.05 ± 3.9	72.4 ± 2.2	82.0 ± 1.6	84.8 ± 1.1
Tox21 (AUC %)	80.7 ± NA	75.0 ± 0.8	80.1 ± 3.9	81.43 ± 2.3	**85.1 ± 2.1**	84.8 ± 1.7	82.3 ± 2.3

in the node-level module. In Table 4, we used GIN (Xu et al., 2019) to show the flexibility of GiG architecture and its possibility to adapt to the input data. For classification losses, optimizer, schedulers, node-level layers, and pooling operators, we used the versions implemented in PyTorch (Paszke et al., 2019) and PyTorch Geometric (Fey and Lenssen, 2019). For clustering techniques, we used the implementation from scikit-learn (Pedregosa et al., 2011). All models were trained and tested on workstation hardware (Intel 6-core i7 CPU, 16 GB RAM, NVIDIA GeForce RTX 2060 GPU, 6 GB VRAM). The source code will be available online [2].

Dataset split. The HCP dataset is split into training (72%), validation (8%), and test (20%) sets (Pervaiz et al., 2020). Results are averaged over five consecutive runs of the best model. For $PROTEINS_{29}$, we follow (Zhang et al., 2019), and randomly split the dataset for train and test sets with 90% vs. 10% and repeated it 10 times. For each run, the test set is fixed first, then the training set is split to train and validation sets by the k-folds split, where k=10. The final evaluation was based on 10 runs. For the Tox21 dataset, we use predefined scaffold splits from (Hu et al., 2020). The same sets were used for all the GiG models and baselines. For a fair comparison, in D&D, NCI1, ENZYMES, and $PROTEINS_3$ we use the same test protocol as in (Errica et al., 2020).

Importance of batch size. Our method does not require the full population to be known a priori. Instead, it builds the population graph on a subset of samples according to the extracted batch. This has the advantage of both reducing the computational complexity and allowing our method to be used in the more challenging inductive setting, where part of the population is unknown at the training time. As a consequence, the number of input graphs that we consider in a batch has a direct effect on the performance of our method. To investigate this aspect, we fixed the batch size to a specific value and optimized the remaining hyper-parameters according to the validation loss. In Fig. 4 (blue color), we report results obtained on the $PROTEINS_3$ dataset using our LGL+NDD model. Small triangles indicate mean accuracy and tails show the standard deviation. From the plot, we can see that considering more graphs in the population is beneficial until up to 50 samples, after which the performance stabilizes. We also investigate the extreme cases of using batch size equal to 1 and batch size equal to the full test population (125 in $PROTEINS_3$) during test time. To show the test set with a bigger population, we experimented with the NCI1 dataset. The best-performing batch size for NCI1 is equal to 75. The full test set = 412. The results (acc ± std) are 59.48 ± 4.5, 81.80 ± 1.8, 80.66 ± 2.9 for batch sizes equal to 1, 75, and 412, respectively, which confirms the trend we observe in the $PROTEINS_3$ dataset. From Fig. 4, it is clear that performance drops with smaller batch sizes. To mitigate this effect and address the problem that an ideal amount of objects might not be available during test time, we also suggest combining the test samples with training samples. With this combination, it would still be possible to achieve the best-performing batch-size results. In Fig. 4, we show experiments where the population graph was predicted on a combination of test and training samples. While the total batch size is fixed to 100 we let the number of test samples in the combination grow from 1 to 100. Accuracy was computed only on the test samples and final results were computed on 10 folds as in the other ablation experiments.

[2] https://github.com/mullakaeva/GiG

Table 4. Comparison with general purpose graph classification methods : Deep-Graph-CNN (Zhang et al., 2018), DiffPool (Ying et al., 2018), ECC (Simonovsky and Komodakis, 2017), GIN (Xu et al., 2019), GraphSAGE (Hamilton et al., 2017). Competitive methods numbers were taken from (Errica et al., 2020). Top three performances are in bold, blue, and violet.

	D&D	NCI1	$PROTEINS_3$	ENZYMES
Deep-Graph-CNN	76.6 ± 4.3	76.4 ± 1.7	72.9 ± 3.5	38.9 ± 5.7
DiffPool	75.0 ± 3.5	76.9 ± 1.9	73.7 ± 3.5	59.5 ± 5.6
ECC	72.6 ± 4.1	76.2 ± 1.4	72.3 ± 3.4	29.5 ± 8.2
GIN	75.3 ± 2.9	80.0 ± 1.4	73.3 ± 4.0	**59.6 ± 4.5**
GraphSAGE	72.9 ± 2.0	76.0 ± 1.8	73.0 ± 4.5	58.2 ± 6.0
GiG DGCNN	70.4 ± 5.9	79.1 ± 2.2	70.8 ± 4.0	30.3 ± 8.7
GiG LGL	74.9 ± 3.9	81.0 ± 1.9	75.0 ± 2.7	54.5 ± 5.6
GiG LGL+NDD	**76.7 ± 4.9**	**81.8 ± 1.8**	**75.6 ± 3.9**	50.0 ± 7.0

Table 5. Comparison of recent methods on HCP dataset: ElasticNet (Pervaiz et al., 2020) DECENNT(Mahmood et al., 2022) ST-GCN (Gadgil et al., 2020) rs-fMRI-GIN (Kim and Ye, 2020). "G", "S", and "M" indicate respectively the GroupICA (Smith et al., 2014; Beckmann and Smith, 2004), Shaefer (Schaefer et al., 2018) and multi-modal (Glasser et al., 2016) parcellation approaches with the corresponding number of parcels.

Study	GiG LGL+NDD	ElasticNet	DECENNT	ST-GCN	rs-fMRI-GIN
acc ± std	89.7 ± 1.3	85.5 ± 2.0	86.0 ± NA	83.7 ± 3.5	84.6 ± 2.9
Parcellation	G 200	G 200	S 200	S 400	M 22
Subjects	1003	1003	942	942	1091

4.3. Quantitative results

This subsection shows experiments on three main and four benchmark datasets. To show the benefits of adding the population-level module, we compared our models with a regular graph convolutional baseline (GCN in Table 3) , consisting of the same node-level module $F1$ and classifier as GiG

Table 6. Evaluation of different k for building the graph in the KNN model. The top performance is in bold.

dataset	5	6	7	9	10	15	20	25	35
HCP	85.42 ± 8.14	83.6 ± 10.3	81.96 ± 6.5	**85.87± 2.6**	76.98 ± 6.7	75.11 ± 13.01	75.28 ± 6.2	48.8 ± 5.2	50.93 ± 8.04
$PROTEINS_{29}$	74.41 ± 5.3	74.53 ± 6.8	**75.05 ± 3.9**	73.91 ± 7.3	74.26 ± 7.7	72.48 ± 12.5	70.41 ± 5.9	71.85 ± 3.6	72.9 ± 11.9
Tox21	81.28 ± 3.11	**81.45 ± 1.49**	81.43 ± 2.38	73.51 ± 7.5	80.91 ± 1.0	74.74 ± 13.87	79.91± 1.2	79.38 ± 2.1	64.82 ± 0.12

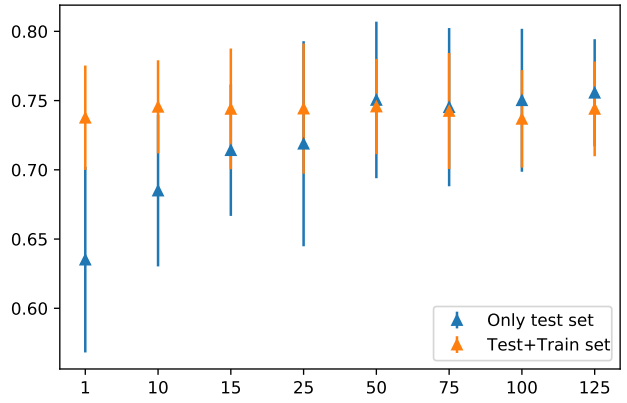
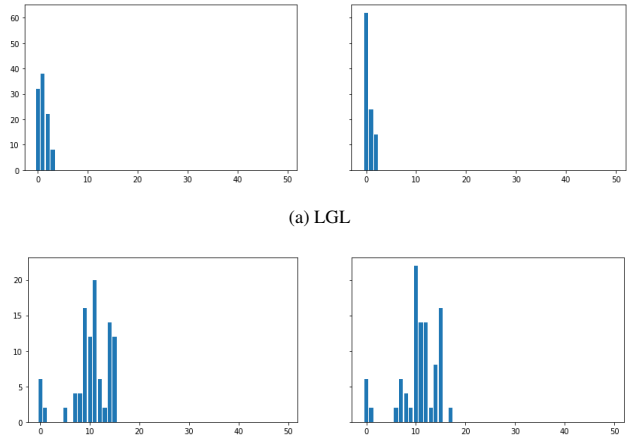


Fig. 4. $PROTEINS_3$. The x-axis shows the amount of test set (blue color) in combination with training set objects (orange color). Triangles indicate mean accuracy and tails standard deviation.



(a) LGL

(b) LGL+NDD

Fig. 5. $PROTEINS_{29}$ GiG LGL vs GiG LGL+NDD node degree distributions comparison. The x-axis shows node degrees divided into 50 bins, the y-axis shows the occurrence of nodes degree. The left column shows the "soft" node degree distribution before thresholding, and the right column after applying a threshold of 0.5.

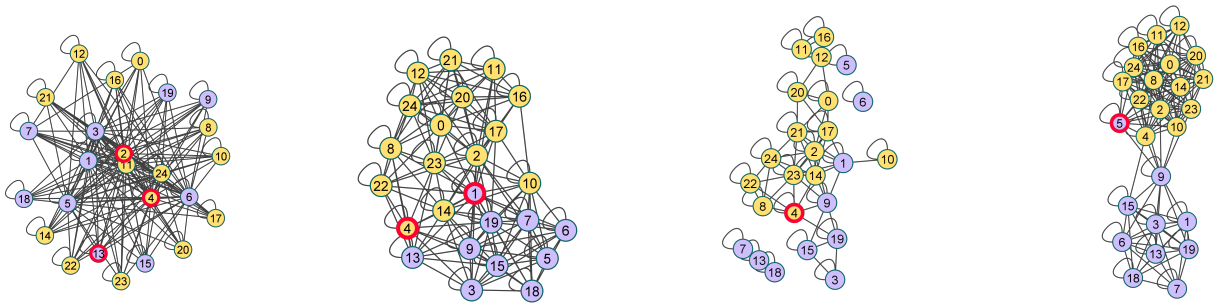
models (details in Table 10), and a state-of-the-art or classical method for each application: HGP-SL (Zhang et al., 2019) for proteins, DECENNT (Mahmood et al., 2022) for HCP, and ToxicBlend (Zaslavskiy et al., 2018) for Tox21. Further, we investigate three different baseline methods of building the population graph. First, to show the benefit of learning the graph, we suggest building the population-level graph sampling a random graph using Erdős–Rényi model (Erdos et al., 1960) (Random). The graph is constructed by connecting nodes randomly with a probability $p = 0.1$. Each edge can be chosen with equal probability and independently of the others. The number of nodes corresponds to the batch size. The second strategy consists of building a KNN graph based on the input graph similarities computed with the WL graph kernel (Shervashidze et al., 2011), where the node degree k is optimized for each dataset (KNN). Lastly, as a hybrid approach, we adopt a strategy similar to DGCNN (Wang et al., 2019) and dynamically build a KNN graph according to the output features similarity of the first module F_1 (GiG DGCNN). More precisely, the population-level module of GiG DGCNN consists of DynamicEdgeConv layer (Wang et al., 2019) applied on the output of the linear layer and ReLU (Agarap, 2018) activation function (Table 10). Despite changing during the training process, with this approach, the graph structure is not directly learned as in our method. For each baseline, the node-level module $F1$ and the classifier remain the same.

HCP. Table 3 row 1 shows the results for experiments on sex prediction task on the HCP dataset. We compare our results with the state-of-the-art DECENNT model (Mahmood et al., 2022) and the classical ElasticNet (not graph-based) model (Pervaiz et al., 2020). The proposed models LGL and LGL+NDD outperform DECENNT and ElasticNet by 3.4% , 3.7 % and 3.9% , 4.2 %, respectively. The superiority of our GiG models is consistent in comparison to other recent models tailored to the HCP dataset as well, Table 5. Further, comparing GCN with population-graph-based models (DGCNN, LGL, LGL+NDD), we see that a proper population graph helps with the downstream task and boosts the performance by at least 4.78 %. In contrast, the comparison with fixed Random population graph using GCN shows that the wrong population graph might significantly decrease the final classification performance.

$PROTEINS_{29}$. In the $PROTEINS_{29}$ dataset (Table 3 row 2), the LGL+NDD model performs very close to the classical model HGP-SL (Zhang et al., 2019). The difference in accuracy is 0.13 %-points, however, LGL+NDD is more stable among the folds and the final standard error is 50% lower in comparison with HGP-SL. LGL and LGL+NDD exceed the GCN by 2 and 4.8 %-points. This indicates that among graph-based approaches, learning the population-level graph can help with the downstream task. Among all suggested models, LGL+NDD gives the best results (84.8 ± 1.08), which is higher than LGL by 2.8 %-points. Regularizing node degrees with the NDD loss stabilizes the learning process and boosts classification performance.

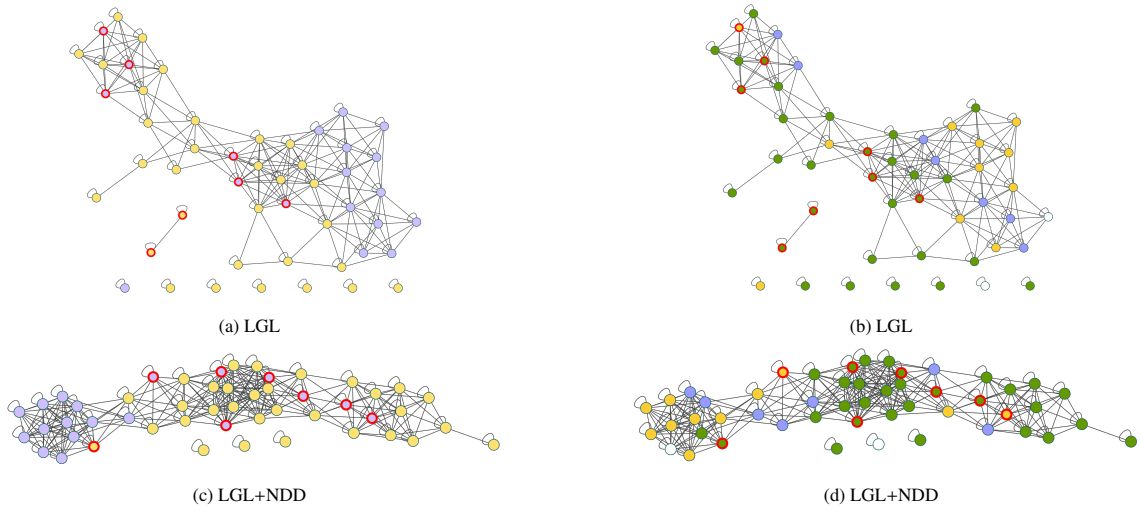
Tox21. For the Tox21 dataset (Table 3 row 3), Random and KNN baselines perform similarly and are close to the ToxicBlend results on scaffold split (Zaslavskiy et al., 2018). LGL+NDD and LGL outperform ToxicBlend (Zaslavskiy et al., 2018) with a gap of 4.97 and 7.5 %-points. The performance difference between LGL and DGCNN is not significant (t-Test with the p-value = 0.7483).

Bench-marking datasets. The GiG LGL+NDD model out-



(a) KNN (b) GiG DGCNN (c) LGL (d) LGL+NDD

Fig. 6. HCP: population graphs comparison according to misclassified identifiers. The location of the node depends purely on the discrete structure of the graph. Misclassified nodes are circled in red.



(a) LGL (b) LGL

(c) LGL+NDD (d) LGL+NDD

Fig. 7. PROTEINS$_{29}$: population graphs comparison. Misclassified nodes are circled in red. The plotting threshold for LGL is equal to 0.01 and for LGL+NDD is 0.5 respectively, which means that only edges with weight values bigger than the chosen threshold were plotted. Colors in (a),(c) represent two classes: 0 (yellow) and 1 (violet). Colors in (b) and (d) represent the CATH hierarchy: yellow, violet, and green correspond to mainly alpha, mainly beta, and alpha-beta respectively.

performs the SOTA classification models for D&D, NCI1, and PROTEINS$_3$ dataset, with a margin of 0.1, 1.8, 1.9 % respectively (Table 4). For NCI1 and PROTEINS$_3$ datasets LGL is the second-best-performing model.

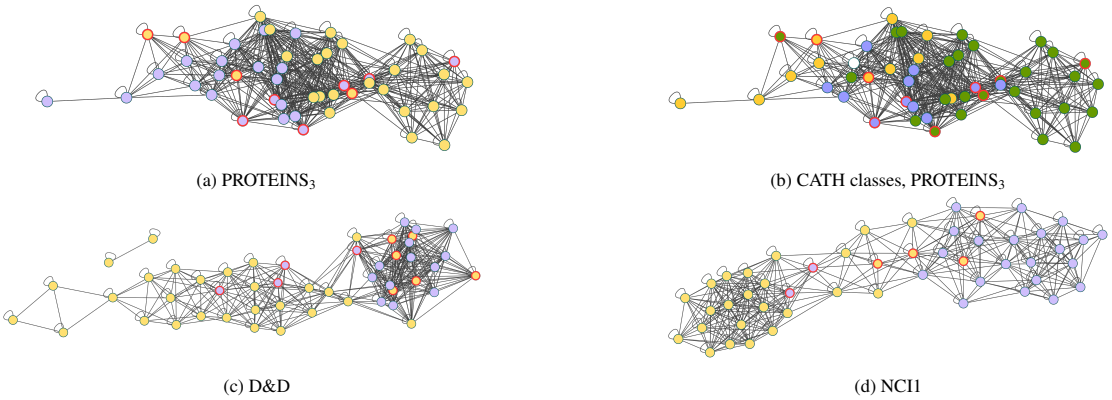
4.4. Knowledge discovery analysis

In this section, we show and discuss the population-level graphs and how they might be exploited as a source of additional information for data exploration.

HCP: As can be seen in Fig. 6a, all presented methods have similar performance. However, it is hard to see clusters in the KNN baseline since all objects are connected and grouped. In comparison, all population graphs from the proposed GiG methods are well clustered. Importantly, the LGL adjacency matrix $\mathbf{A_p}$ of the learned population graph consists entirely of near-zero values, indicating that the model is not confident about the learned graph structure. For interpretability analysis, we plotted LGL in Fig. 6c with an edge weight threshold at a level of 1e−1. In contrast, the NDDL regularization results in well-binarized edge weights, therefore we opted for a threshold of 0.5. Fig. 6 shows the unique numbers of each node as well. DGCNN locates nodes number "4" and "1" on the border of two classes and misclassifies them. LGL+NDD mislocates node "5", which, in the previous population graphs, was located inside its cluster. For all methods except KNN, some nodes tend to be located together, e.g. nodes with identifiers "11", "12" and "16" as well as "23", "2", "14". This is an indicator that these nodes could have similar properties or common characteristics.

PROTEINS$_{29}$: In Fig. 7a and Fig. 7c, GiG LGL and GiG LGL+NDD show a clustering trend. However, similar to the HCP dataset, the LGL model is quite uncertain about the learned graph connections, as we can see in Fig. 2, where most adjacency matrix values are located near-zero for LGL. In contrast, using the NDDL, the GiG LGL+NDD model is more confident regarding its decision (Fig. 2b). Also, looking at Fig. 5, we notice that the binarized node degree distribution of GiG LGL is skewed to the left, while the one of the LGL+NDD model remains almost the same after the thresholding, which also indicates the higher confidence of the model benefiting from NDDL regularization.

In Figs. 7b and (7d), we see that nodes are clustered ac-



(a) $PROTEINS_3$ (b) CATH classes, $PROTEINS_3$

(c) D&D (d) NCI1

Fig. 8. Population graphs produces by the GiG LGL+NDD model. Misclassified nodes are circled in red. The plotting threshold for LGL+NDD is 0.5, which means that only edges with weight values bigger than the chosen threshold were plotted. Colors in (a),(c), and (d) represent two classes: 0 (yellow) and 1 (violet). Colors in (b) represent the CATH hierarchy: yellow, violet, and green correspond to mainly alpha, mainly beta, and alpha-beta respectively.

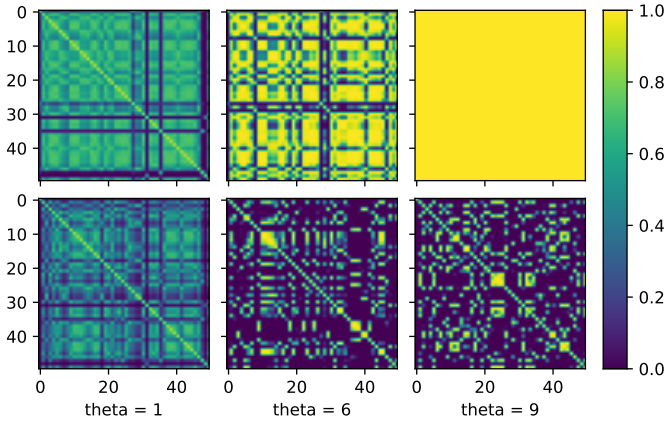


Fig. 9. Influence of θ on population the graphs of GiG LGL and LGL+NDD models on $PROTEINS_3$ dataset. The first and second rows refer to LGL and LGL+NDD models respectively.

cording to CATH classes: "1", "2", "3", i.e. "mainly alpha", "mainly beta" and "alpha-beta" respectively. Remarkably, most nodes that were misclassified according to their task labels are located on the proper CATH class clusters. This indicates that GiG models not only cluster population-level graphs according to training labels, but also capture the structure and properties of the input data. Moreover, the majority of prediction errors concern the proteins that belong to alpha-beta CATH classes. This is explainable with domain knowledge: "alpha-beta" protein structures are composed of α-helices and β-strands, which are combined inside the structure of "mainly alpha" or "mainly beta" proteins. Therefore, it is more challenging for the model to find a proper location inside the population graph for these input objects.

Tox21: Tox21 is a multi-task dataset, where assays are highly correlated with each other, and training a model towards only one of the toxic tests is less effective (Jiang et al., 2021). On the other hand, analyzing the population graph with multiple labels is challenging, and we thus report only quantitative results for the Tox21 dataset. Once again, we observe that incorporating the population graph helped improve the final downstream task performance.

Bench-marking datasets: The population-level graphs from GiG LGL+NDD for NCI1, D&D and $PROTEINS_3$ are shown in Fig. 8. The misclassification of some nodes is to be expected, given the structure of the $\mathbf{G_p}$ graph. The node representations might be misleading for node embeddings obtained via GCN-based models, given their wrong neighborhood associations. Explaining these locations requires some additional information, like the CATH hierarchy for PROTEINS dataset. From the $PROTEINS_3$ population graphs, it is evident that most misclassified nodes are embedded at locations that correspond to the CATH classes. This indicates that population graphs and their node embedding provide a form of knowledge discovery that can be interpreted with additional domain information of the input objects.

Population graph evaluation using clustering methods. To perform quantitative analyses of the learned $\mathbf{G_p}$, we evaluated the population graph structure on benchmark datasets using unsupervised node clustering techniques: Spectral Clustering (Von Luxburg, 2007) and K-means (Lloyd, 1982). Performance metrics (accuracy ± standard deviation) are computed between ground truth labels and predicted clustering labels (Table 8). For the CATH hierarchy, we compared CATH classes with clustering labels. Since CATH labels are only used for knowledge discovery, we cannot compute any performance metric. The accuracy scores obtained by clustering the learned latent graph show that our method is indeed capturing relations relevant to the classification.

Table 7. Learned μ and σ for different datasets. $PROTEINS_{29}$, HCP and Tox21 were initialized with μ=7.0 and σ=2.0, $PROTEINS_3$, NCI1, D&D, ENZYMES with μ=9.0 and σ=3.0.

	μ	σ
HCP	5.40 ± 0.78	3.69 ± 0.38
$PROTEINS_{29}$	9.80 ± 1.30	7.30 ± 0.48
Tox21	7.60 ± 1.20	5.07 ± 1.76
$PROTEINS_3$	15.12 ± 6.09	8.93 ± 3.84
NCI1	10.68 ± 3.34	7.14 ± 2.56
D&D	9.00 ± 0.01	3.00 ± 0.01
ENZYMES	1.34 ± 3.00	0.94 ± 0.27

Table 8. Population graph evaluation using clustering methods (acc±std).

	GiG LGL+NDD	Spectral clustering	K-means
$PROTEINS_3$	75.6 ± 3.9	59.96 ± 3.82	66.54 ± 3.21
NCI1	81.8 ± 1.8	62.22 ± 7.43	63.29 ± 6.54
D&D	76.7 ± 4.9	63.60 ± 7.34	63.97 ± 8.22
$PROTEINS_3$ CATH	-	70.41 ± 6.24	69.80 ± 7.59

Table 9. Evaluation of learned θ and corresponding models (LGL, LGL+NDD) performance (acc±std) for different θ intilizations. The best-performing cases are shown in bold.

	LGL		LGL+NDD	
θ	learned θ	acc±std	learned θ	acc±std
0.0	0.52 ± 0.28	69.09 ± 5.01	-0.09 ± 0.35	67.50 ± 8.09
0.5	1.22 ± 0.32	68.76 ± 5.68	0.46 ± 0.03	74.20 ± 3.92
1.0	**0.94 ± 0.10**	**73.96 ± 4.28**	**1.00 ± 0.05**	**75.07 ± 4.28**
2.0	1.91 ± 0.16	73.15 ± 4.65	0.87 ± 0.89	73.29 ± 3.06
3.0	3.00 ± 0.08	73.68 ± 4.62	2.53 ± 0.48	73.87 ± 2.42
4.0	4.02 ± 0.18	72.41 ± 3.29	3.91 ± 0.05	73.16 ± 3.16
5.0	5.03 ± 0.13	70.82 ± 3.85	5.06 ± 0.17	71.00 ± 6.1
6.0	6.11 ± 0.23	73.69 ± 3.76	**5.54 ± 0.35**	**75.20 ± 3.82**
7.0	6.98 ± 0.13	73.79 ± 3.82	6.46 ± 0.32	73.31 ± 4.23
8.0	7.98 ± 0.16	72.65 ± 5.12	**7.73 ± 0.05**	**75.40 ± 2.44**
9.0	**8.96 ± 0.13**	**74.04 ± 5.8**	8.97 ± 0.10	74.67 ± 4.44
10.0	9.98 ± 0.10	70.16 ± 4.76	8.02 ± 0.04	74.45 ± 2.82
11.0	10.95 ± 0.1	73.51 ± 4.63	11.05 ± 0.07	67.63 ± 5.85
12.0	11.94 ± 0.14	73.04 ± 4.64	12.10 ± 0.10	65.13 ± 6.56

5. Discussion

To learn the population graph in LGL-based models, we utilize a soft threshold that can be tuned by two learnable parameters: *temp* and θ. One future direction could be to learn the θ values individually for each node. The latter hugely affects the adjacency matrix values and might help to push them further to either 0 or 1. Even though these parameters are learned, the proper initialization is crucial and might lead to different population graphs and different classification performances (Table 9). The influence of θ thresholds on GiG models is not consistent. Increasing the θ in the GiG LGL model mostly leads to graphs with adjacency matrix values close to 1. The NDDL regularization mitigates this issue by providing consistently well-structured graphs (Fig. 9).

In our experiments, we also observe that obtaining proper input-graph representations is an important factor since the population graph is learned or constructed based on them. The node-level module and pooling operator should be adapted to the application and the input data.

Another crucial point of our method is the learnable target distribution via NDDL in GiG LGL+NDD. The NDDL directly affects the structure of the predicted population graph, and its contribution is twofold: It helps in improving the interpretability of the learned population graph and, most of the times, leads to better performance on the downstream task. In this work, as a starting point, we have chosen a normal distribution as a generic target shape to regularize the latent graph degrees. The concrete shape is data driven, as the actual parameters μ and σ of the normal distribution are optimized during training, which allows the model to tune them according to the specific dataset/task. Table 7 shows that for the different tasks the model learns different parameters. Nevertheless, the unknown edge probability distribution of the population graph is generally complex and unknown, given that the medical data is collected from a cohort of subjects. Directions for future work include the investigation of the influence of different target distributions and their reasoning, as well as considering graph pooling operators. In addition, all constructed and learned latent graphs had been established based on similarities between input objects. However, one more relevant and challenging application for this work might be learning the graph representing Protein-protein interactions (PPI).

The main limitations of the proposed method, which we aim to address in the future, can thus be identified in the following two points. 1) The choice of the target distribution in NDDL needs further exploration. In this paper, we only considered Gaussian distribution. 2) Even though the model gives us the freedom of choice for F_1 and F_2, a careful selection of F_1 and F_2 is necessary.

6. Conclusion

In this paper, we propose a method for learning a latent graph structure on graph-valued input data. The latent graph provides a form of knowledge discovery on the distribution and neighborhood relationships between input samples. Further, the latent graph is learned end-to-end, along with the downstream task, e.g. classification. More precisely, we propose an architecture consisting of three main modules: node-level, population-level, and GCN classifier. Our architecture can be easily adapted to any graph classification application. Our experimental setup shows that learning the population graph increases the performance of the downstream task for most of the datasets (HCP, Tox21, NCI1, D&D, $PROTEINS_3$ datasets). Additionally, the latent graph might be used to understand the decision taken by the model (PROTEINS) and discover new information based on interconnected objects. Moreover, we introduced a Node Degree Distribution Loss (NDDL) based on Kullback–Leibler divergence regularising the latent graph's node degree distribution, resulting in a better representation of the graph-valued input population.

Acknowledgments

Kamilia Zaripova and Nassir Navab report financial support was provided by the Federal Ministry of Education and Research of Germany (BMBF) under project DIVA (FKZ 13GW0469C). Luca Cosmo reports financial support was provided by IRIDE project from DAIS departement of Ca' Foscari University of Venice. Anees Kazi reports financial support was provided by TUM-ICL incentive funding 2019.

References

Agarap, A.F., 2018. Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375 .

Table 10. Hyperparameters optimized for Table 3. "bs", "k" and "S" indicate batch size, number of k in the KNN graph, and scheduler. DECL stands for DynamicEdgeConv(ReLu(Linear(2*-,-))). "P" and "C" indicate *Reduce Learning Rate on Plateau* and *Cosine Annealing*. F_1, F_3 GNN, and classifier are the same for all the GiG and baseline experiments within the dataset, except for the GCN model that does not require F_3 GNN. "-" indicates the dimension from the previous or subsequent layer.

	bs	F_1 + pooling	F_2 (LGL)	F_2 (DGCNN)	k	F_3 GNN	F_3 Classifier	S	α	t	θ	lr_θ_t	lr_μ_σ
HCP	25	GraphConv(-,100) GraphConv(64) Pooling(mean)	Linear(-,64) Linear(40)	DECL(2*-,64)	9	GraphConv(-,36) GraphConv(24)	Linear(-,20) Linear(16) Linear(10) Linear(2)	C	1.0	1e-4	1.0	1e-2	1e-2
$PROTEINS_{29}$	50	GraphConv(-,30) GraphConv(28) Pooling(mean)	Linear(-, 24) Linear(20)	DECL(2*-,128)	7	GraphConv(-,20)	Linear(-,20) Linear(16) Linear(12) Linear(2)	P	3e-3	1e-1	1.0	1e-3	1e-1
Tox21	50	GraphConv(-,30) GraphConv(28) Pooling(add)	Linear(-, 24) Linear(20)	DECL(2*-,24)	6	GraphConv(-,20)	Linear(-,20) Linear(16) Linear(12) Linear(12)	P	1e-4	1e-1	8.0	1e-3	1e-3

Table 11. Hyperparameters used for model selection for Table 4 using Optuna framework (Akiba et al., 2019). Square brackets indicate the range for each parameter. The number of $F1$ layers and t are fixed and equal to 2 and 1e-4 respectively. We used Reduce Learning Rate on Plateau as a scheduler for all the datasets in the current table. $F2$ (LGL, LGL+NDD) layers and $F2$ (DGCNN) layers are fixed and are the same for all the datasets and equal to linear layers with dimensions (-,24,20) with a ReLU activation function in between and DECL layer - DynamicEdgeConv(ReLU(Linear(2*-, 128))) respectively. The classifier consists of linear layers (-, 16, 12,-). "-" indicates the dimension from the previous or subsequent layer.

	batch size	F1 dims	pooling	#GNN layers	GNN dims	opt_lr	α	θ	lr_θ_t	lr_μ_σ
$PROTEINS_3$	[25, 125]	[-, 300]	[mean, add]	[2, 5]	[20, 120]	[1e-4, 1e-1]	[1e-3, 3e-1]	[0.5, 11]	1e-2	1e-2
D&D									1e-3	1e-4
NCI1									1e-2	1e-2
ENZYMES									1e-2	1e-2

Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M., 2019. Optuna: A next-generation hyperparameter optimization framework, in: Teredesai, A., Kumar, V., Li, Y., Rosales, R., Terzi, E., Karypis, G. (Eds.), International Conference on Knowledge Discovery and Data Mining, ACM. pp. 2623–2631. doi:10.1145/3292500.3330701.

Baek, J., Kang, M., Hwang, S.J., 2021. Accurate learning of graph representations with graph multiset pooling, in: International Conference on Learning Representations, OpenReview.net.

Barta, G., 2016. Identifying biological pathway interrupting toxins using multi-tree ensembles. Frontiers in Environmental Science 4. doi:10.3389/fenvs.2016.00052.

Beckmann, C.F., Smith, S.M., 2004. Probabilistic independent component analysis for functional magnetic resonance imaging. IEEE transactions on medical imaging 23, 137–152. doi:10.1109/tmi.2003.822821.

Bessadok, A., Mahjoub, M.A., Rekik, I., 2019a. Hierarchical adversarial connectomic domain alignment for target brain graph prediction and classification from a source graph, in: Rekik, I., Adeli, E., Park, S.H. (Eds.), Predictive Intelligence in Medicine, Springer International Publishing, Cham. pp. 105–114. doi:10.1007/978-3-030-32281-6_11.

Bessadok, A., Mahjoub, M.A., Rekik, I., 2019b. Symmetric dual adversarial connectomic domain alignment for predicting isomorphic brain graph from a baseline graph, in: Shen, D., Liu, T., Peters, T.M., Staib, L.H., Essert, C., Zhou, S., Yap, P.T., Khan, A. (Eds.), Medical Image Computing and Computer Assisted Intervention, Springer International Publishing, Cham. pp. 465–474. doi:10.1007/978-3-030-32251-9_51.

Bessadok, A., Mahjoub, M.A., Rekik, I., 2021. Brain graph synthesis by dual adversarial domain alignment and target graph prediction from a source graph. Medical Image Analysis 68, 101902. doi:10.1016/j.media.2020.101902.

Bicciato, A., Cosmo, L., Minello, G., Rossi, L., Torsello, A., 2023. Classifying me softly: A novel graph neural network based on features soft-alignment, in: Krzyzak, A., Suen, C.Y., Torsello, A., Nobile, N. (Eds.), Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops, Springer. Springer International Publishing. pp. 43–53. doi:10.1007/978-3-031-23028-8_5.

Borgwardt, K.M., Ong, C.S., Schönauer, S., Vishwanathan, S.V.N., Smola, A.J., Kriegel, H.P., 2005. Protein function prediction via graph kernels. Bioinformatics 21, i47–i56. doi:10.1093/bioinformatics/bti1007, arXiv:https://pubmed.ncbi.nlm.nih.gov/15961493/.

Bouritsas, G., Loukas, A., Karalias, N., Bronstein, M., 2021. Partition and code: learning how to compress graphs. Advances in Neural Information Processing Systems 34, 18603–18619.

Burwinkel, H., Kazi, A., Vivar, G., Albarqouni, S., Zahnd, G., Navab, N., Ahmadi, S.A., 2019. Adaptive image-feature learning for disease classification using inductive graph networks, in: Medical Image Computing and Computer Assisted Intervention, Springer International Publishing, Cham. pp. 640–648.

Chaari, N., Gharsallaoui, M.A., Akdağ, H.C., Rekik, I., 2022. Multigraph classification using learnable integration network with application to gender fingerprinting. Neural Networks 151, 250–263. doi:10.1016/j.neunet.2022.03.035.

Choma, N., et al., 2018. Graph neural networks for icecube signal classification, in: Wani, M.A., Kantardzic, M., Mouchaweh, M.S., Gama, J., Lughofer, E. (Eds.), International Conference on Machine Learning, IEEE. pp. 386–391. doi:10.1109/icmla.2018.00064, arXiv:1809.06166.

Cortes, C., Vapnik, V., 1995. Support vector networks. Machine Learning 20, 273–297. doi:10.1007/bf00994018.

Cosmo, L., Kazi, A., Ahmadi, S.A., Navab, N., Bronstein, M.M., 2020. Latent-graph learning for disease prediction, in: Martel, A.L., Abolmaesumi, P., Stoyanov, D., Mateus, D., Zuluaga, M.A., Zhou, S.K., Racoceanu, D., Joskowicz, L. (Eds.), Medical Image Computing and Computer Assisted Intervention, Springer. pp. 643–653. doi:10.1007/978-3-030-59713-9_62, arXiv:2003.13620.

Cosmo, L., Minello, G., Bronstein, M.M., Rodolà, E., Rossi, L., Torsello, A., 2021. Graph kernel neural networks. CoRR abs/2112.07436. doi:10.48550/ARXIV.2112.07436, arXiv:2112.07436.

Dash, S., Shakyawar, S., Sharma, M., Kaushik, S., 2019. Big data in healthcare: management, analysis and future prospects. Journal of Big Data 6, 54. doi:10.1186/s40537-019-0217-0.

Demir, U., Gharsallaoui, M., Rekik, I., 2020. Clustering-Based Deep Brain MultiGraph Integrator Network for Learning Connectional Brain Templates. Springer International Publishing. volume 12443. pp. 109–120. doi:10.1007/978-3-030-60365-6_11.

Dobson, P.D., Doig, A.J., 2003. Distinguishing enzyme structures from non-enzymes without alignments. Journal of Molecular Biology 330, 771–783. doi:10.1016/S0022-2836(03)00628-4.

Duvenaud, D.K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., Adams, R.P., 2015. Convolutional networks on graphs for learning molecular fingerprints, in: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (Eds.), Annual Conference on Neural Information Processing Systems, pp. 2224–2232.

Erdos, P., Rényi, A., et al., 1960. On the evolution of random graphs. Publ. Math. Inst. Hung. Acad. Sci 5, 17–60. doi:10.1515/9781400841356.38.

Errica, F., Podda, M., Bacciu, D., Micheli, A., 2020. A fair comparison of graph neural networks for graph classification, in: International Conference on Learning Representations, OpenReview.net.

Essen, D.C.V., Smith, S.M., Barch, D.M., Behrens, T.E., Yacoub, E., Ugurbil, K., 2013. The wu-minn human connectome project: An overview. NeuroImage 80, 62–79. doi:10.1016/j.neuroimage.2013.05.041. mapping the Connectome.

Fey, M., Lenssen, J.E., 2019. Fast graph representation learning with pytorch geometric, in: International Conference on Learning Representations, Workshop on Representation Learning on Graphs and Manifolds. arXiv:1903.02428.

Fung, V., Zhang, J., Juarez, E., Sumpter, B.G., 2021. Benchmarking graph neural networks for materials chemistry. npj Computational Materials 7, 1–8. doi:10.26434/chemrxiv.13615421.v2.

Gadgil, S., Zhao, Q., Pfefferbaum, A., Sullivan, E.V., Adeli, E., Pohl, K.M., 2020. Spatio-temporal graph convolution for resting-state fmri analysis, in: Martel, A.L., Abolmaesumi, P., Stoyanov, D., Mateus, D., Zuluaga, M.A., Zhou, S.K., Racoceanu, D., Joskowicz, L. (Eds.), International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer. Springer International Publishing. pp. 528–538. doi:10.1007/978-3-030-59728-3_52.

Gainza, P., Sverrisson, F., Monti, F., Rodola, E., Bronstein, M.M., Correia, B.E., 2019. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. Nature Methods 17, 184–192. doi:10.1038/s41592-019-0666-6.

Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E., 2017. Neural message passing for quantum chemistry, in: Precup, D., Teh, Y.W. (Eds.), International Conference on Machine Learning, JMLR.org. pp. 1263–1272.

Glasser, M.F., Coalson, T.S., Robinson, E.C., Hacker, C.D., Harwell, J., Yacoub, E., Ugurbil, K., Andersson, J., Beckmann, C.F., Jenkinson, M., et al., 2016. A multi-modal parcellation of human cerebral cortex. Nature 536, 171–178. doi:10.1038/nature18933.

Guo, Z., Yu, W., Zhang, C., Jiang, M., Chawla, N.V., 2020. GraSeq: Graph and Sequence Fusion Learning for Molecular Property Prediction. Association for Computing Machinery, New York, NY, USA. pp. 435–443. doi:10.1145/3340531.3411981.

Guo, Z., Zhang, C., Yu, W., Herr, J., Wiest, O., Jiang, M., Chawla, N.V., 2021. Few-shot graph learning for molecular property prediction, in: Leskovec, J., Grobelnik, M., Najork, M., 0001, J.T., Zia, L. (Eds.), Web Conference, ACM. pp. 2559–2567. doi:10.1145/3442381.3450112, arXiv:2102.07916.

Gurbuz, M.B., Rekik, I., 2020. Deep graph normalizer: A geometric deep learning approach for estimating connectional brain templates, in: Martel, A.L., Abolmaesumi, P., Stoyanov, D., Mateus, D., Zuluaga, M.A., Zhou, S.K., Racoceanu, D., Joskowicz, L. (Eds.), Medical Image Computing and Computer Assisted Intervention. Springer International Publishing. volume 12267, pp. 155–165. doi:10.1007/978-3-030-59728-3_16, arXiv:2012.14131.

Gürbüz, M.B., Rekik, I., 2021. Mgn-net: a multi-view graph normalizer for integrating heterogeneous biological network populations. Medical Image Analysis 71, 102059. doi:10.1016/j.media.2021.102059.

Hamilton, W., Ying, Z., Leskovec, J., 2017. Inductive representation learning on large graphs, in: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Annual Conference on Neural Information Processing Systems, Curran Associates, Inc.. pp. 1024–1034.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J., 2020. Open graph benchmark: Datasets for machine learning on graphs, in: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H.T. (Eds.), Annual Conference on Neural Information Processing Systems.

Huang, W., Zhang, T., Rong, Y., Huang, J., 2018. Adaptive sampling towards fast graph representation learning. Annual Conference on Neural Information Processing Systems 31, 4563–4572.

Idakwo, G., Thangapandian, S., Luttrell, J., Li, Y., Wang, N., Zhou, Z., Hong, H., Yang, B., Zhang, C., Gong, P., 2020. Structure–activity relationship-based chemical classification of highly imbalanced tox21 datasets. Journal of Cheminformatics 12, 66. doi:10.1186/s13321-020-00468-x.

Jiang, B., Zhang, Z., Lin, D., Tang, J., Luo, B., 2019. Semi-supervised learning with graph learning-convolutional networks, in: Conference on Computer Vision and Pattern Recognition, IEEE. pp. 11305–11312. doi:10.1109/CVPR.2019.01157.

Jiang, J., Wang, R., Wei, G.W., 2021. Ggl-tox: Geometric graph learning for toxicity prediction. Journal of Chemical Information and Modeling 61, 1691–1700. doi:10.1021/acs.jcim.0c01294, arXiv:https://doi.org/10.1021/acs.jcim.0c01294. pMID: 33719422.

Kawahara, J., Brown, C.J., Miller, S.P., Booth, B.G., Chau, V., Grunau, R.E., Zwicker, J.G., Hamarneh, G., 2017. Brainnetcnn: Convolutional neural networks for brain networks; towards predicting neurodevelopment. NeuroImage 146, 1038–1049. doi:10.1016/j.neuroimage.2016.09.046.

Kazi, A., Albarqouni, S., Sanchez, A.J., Kirchhoff, S., Biberthaler, P., Navab, N., Mateus, D., 2017. Automatic classification of proximal femur fractures based on attention models, in: 0001, Q.W., Shi, Y., Suk, H.I., Suzuki, K. (Eds.), International Workshop on Machine Learning in Medical Imaging, Springer. Springer International Publishing. pp. 70–78. doi:10.1007/978-3-319-67389-9_9.

Kazi, A., Cosmo, L., Navab, N., Bronstein, M., 2020. Differentiable graph module (dgm) for graph convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022) 45, 1606–1617. doi:10.1109/tpami.2022.3170249, arXiv:2002.04999.

Kazi, A., Shekarforoush, S., Arvind Krishna, S., Burwinkel, H., Vivar, G., Wiestler, B., Kortüm, K., Ahmadi, S.A., Albarqouni, S., Navab, N., 2019a. Graph convolution based attention model for personalized disease prediction, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer. Springer International Publishing. pp. 122–130. doi:10.1007/978-3-030-32251-9_14.

Kazi, A., Shekarforoush, S., Kortuem, K., Albarqouni, S., Navab, N., et al., 2019b. Self-attention equipped graph convolutions for disease prediction, in: International Symposium on Biomedical Imaging, IEEE. IEEE. pp. 1896–1899. doi:10.1109/isbi.2019.8759274, arXiv:1812.09954.

Kazi, A., Shekarforoush, S., Krishna, S.A., Burwinkel, H., Vivar, G., Kortüm, K., Ahmadi, S.A., Albarqouni, S., Navab, N., 2019c. Inceptiongcn: Receptive field aware graph convolutional network for disease prediction, in: Chung, A.C.S., Gee, J.C., Yushkevich, P.A., Bao, S. (Eds.), International Conference on Image Processing and Machine Intelligence, Springer. Springer International Publishing. pp. 73–85. doi:10.1007/978-3-030-20351-1_6, arXiv:1903.04233.

Kersting, K., Kriege, N.M., Morris, C., Mutzel, P., Neumann, M., 2016. Benchmark data sets for graph kernels.

Kim, B.H., Ye, J.C., 2020. Understanding graph isomorphism network for rs-fmri functional connectivity analysis. Frontiers in Neuroscience 14, 630. doi:10.3389/fnins.2020.00630.

Kim, B.H., Ye, J.C., Kim, J.J., 2021. Learning dynamic graph representation of brain connectome with spatio-temporal attention. Advances in Neural Information Processing Systems 34, 4314–4327.

Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization doi:10.48550/ARXIV.1412.6980, arXiv:1412.6980.

Kullback, S., Leibler, R.A., 1951. On information and sufficiency. The annals of mathematical statistics 22, 79–86.

Li, J., Cai, D., He, X., 2017. Learning graph-level representation for drug discovery. arXiv preprint arXiv:1709.03741 abs/1709.03741. arXiv:1709.03741.

Li, R., Wang, S., Zhu, F., Huang, J., 2018. Adaptive graph convolutional neural networks, in: McIlraith, S.A., Weinberger, K.Q. (Eds.), Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, the 30th innovative Applications of Artificial Intelligence, and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI Press. pp. 3546–3553. doi:10.1609/aaai.v32i1.11691.

Li, X., Zhou, Y., Dvornek, N., Zhang, M., Gao, S., Zhuang, J., Scheinost, D., Staib, L.H., Ventola, P., Duncan, J.S., 2021. Braingnn: Interpretable brain graph neural network for fmri analysis. Medical Image Analysis 74, 102233. doi:10.1016/j.media.2021.102233.

Lin, X., Quan, Z., Wang, Z.J., Ma, T., Zeng, X., 2020. Kgnn: Knowledge graph neural network for drug-drug interaction prediction., in: Bessiere, C. (Ed.), International Joint Conferences on Artificial Intelligence, International Joint Conferences on Artificial Intelligence Organization. pp. 2739–2745. doi:10.24963/ijcai.2020/380.

Lloyd, S., 1982. Least squares quantization in pcm. IEEE Transactions on Information Theory 28, 129–137. doi:10.1109/TIT.1982.1056489.

Loshchilov, I., Hutter, F., 2017. Sgdr: Stochastic gradient descent with warm restarts, in: International Conference on Learning Representations, OpenReview.net.

Lu, C., Liu, Q., Wang, C., Huang, Z., Lin, P., He, L., 2019. Molecular property prediction: A multilevel quantum interactions modeling perspective, in: The Thirty-Third AAAI Conference on Artificial Intelligence, The Thirty-First Innovative Applications of Artificial Intelligence Conference, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI Press. pp. 1052–1060. doi:10.1609/aaai.v33i01.33011052.

van der Maaten, L., Hinton, G., 2008. Visualizing data using t-sne. Journal of Machine Learning Research 9, 2579–2605.

Mahmood, U., Fu, Z., Calhoun, V., Plis, S., 2022. Deep dynamic effective connectivity estimation from multivariate time series. arXiv preprint arXiv:2202.02393 , 1–10doi:10.1109/ijcnn55064.2022.9892436, arXiv:2202.02393.

Mansimov, E., Mahmood, O., Kang, S., Cho, K., 2019. Molecular geometry prediction using a deep generative graph neural network. Scientific Reports 9. doi:10.1038/s41598-019-56773-5, arXiv:1904.00314.

Mayr, A., Klambauer, G., Unterthiner, T., Hochreiter, S., 2016. Deeptox: Toxicity prediction using deep learning. Frontiers in Environmental Science 3, 80. doi:10.3389/fenvs.2015.00080.

McInnes, L., Healy, J., Melville, J., 2020. Umap: Uniform manifold approximation and projection for dimension reduction abs/1802.03426. doi:10.48550/ARXIV.1802.03426, arXiv:1802.03426.

Mellema, C., Treacher, A., Nguyen, K., Montillo, A., 2019. Multiple deep learning architectures achieve superior performance diagnosing autism spectrum disorder using features previously extracted from structural and functional mri, in: 2019 IEEE ISBI), IEEE. IEEE. pp. 1891–1895.

Mhiri, I., Nebli, A., Mahjoub, M.A., Rekik, I., 2021. Non-isomorphic inter-modality graph alignment and synthesis for holistic brain mapping, in: Feragen, A., Sommer, S., Schnabel, J.A., Nielsen, M. (Eds.), International Conference on Information Processing in Medical Imaging, Springer. Springer International Publishing. pp. 203–215. doi:10.1007/978-3-030-78191-0_16, arXiv:2107.06281.

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., Bronstein, M.M., 2017. Geometric deep learning on graphs and manifolds using mixture model cnns, in: Conference on Computer Vision and Pattern Recognition, IEEE. pp. 5425–5434. doi:10.1109/cvpr.2017.576, arXiv:1611.08402.

Morris, C., Ritzert, M., Fey, M., Hamilton, W.L., Lenssen, J.E., Rattan, G., Grohe, M., 2019. Weisfeiler and leman go neural: Higher-order graph neural networks, in: The Thirty-Third AAAI Conference on Artificial Intelligence, The Thirty-First Innovative Applications of Artificial Intelligence Conference, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, Association for the Advancement of Artificial Intelligence (AAAI). pp. 4602–4609. doi:10.1609/aaai.v33i01.33014602.

Nebli, A., Gharsallaoui, M.A., Gürler, Z., Rekik, I., 2022. Quantifying the reproducibility of graph neural networks using multigraph data representation. Neural Networks 148, 254–265. doi:10.1016/j.neunet.2022.01.018.

Nikolentzos, G., Vazirgiannis, M., 2020. Random walk graph neural networks, in: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H.T. (Eds.), Annual Conference on Neural Information Processing Systems.

Orengo, C., Michie, A., Jones, S., Jones, D., Swindells, M., Thornton, J., 1997. Cath – a hierarchic classification of protein domain structures. Structure 5, 1093–1109. doi:10.1016/S0969-2126(97)00260-8.

Parisot, S., Ktena, S.I., Ferrante, E., Lee, M., Guerrero, R., Glocker, B., Rueckert, D., 2018. Disease prediction using graph convolutional networks: application to autism spectrum disorder and alzheimer's disease. Medical image analysis 48, 117–130.

Parisot, S., Ktena, S.I., Ferrante, E., Lee, M.C.H., Moreno, R.G., Glocker, B., Rueckert, D., 2017. Spectral graph convolutions for population-based disease prediction, in: Descoteaux, M., Maier-Hein, L., Franz, A.M., Jannin, P., Collins, D.L., Duchesne, S. (Eds.), Medical Image Computing and Computer Assisted Intervention, Springer. pp. 177–185. doi:10.1007/978-3-319-66179-7_21, arXiv:1703.03020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. Pytorch: An imperative style, high-performance deep learning library, in: Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., Garnett, R. (Eds.), Annual Conference on Neural Information Processing Systems, Curran Associates, Inc.. pp. 8024–8035.

Pearl, F., Lee, D., Bray, J., Sillitoe, I., Todd, A., Harrison, A., Thornton, J., Orengo, C., 2000. Assigning genomic sequences to cath. Nucleic acids research 28, 277–82. doi:10.1093/nar/28.1.277.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in python. Journal of Machine Learning Research 12, 2825–2830. arXiv:1201.0490.

Pervaiz, U., Vidaurre, D., Woolrich, M.W., Smith, S.M., 2020. Optimising network modelling methods for fmri. NeuroImage 211, 116604. doi:10.1016/j.neuroimage.2020.116604.

Qi, S., Wang, W., Jia, B., Shen, J., Zhu, S.C., 2018. Learning human-object interactions by graph parsing neural networks, in: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (Eds.), ECCV, Springer International Publishing. pp. 401–417. doi:10.1007/978-3-030-01240-3_25, arXiv:1808.07962.

Qi, X., Liao, R., Jia, J., Fidler, S., Urtasun, R., 2017. 3d graph neural networks for rgbd semantic segmentation, in: International Conference on Computer Vision, IEEE. pp. 5199–5208. doi:10.1109/iccv.2017.556.

Rakhimberdina, Z., Liu, X., Murata, T., 2020. Population graph-based multimodel ensemble method for diagnosing autism spectrum disorder. Sensors 20, 6001. doi:10.3390/s20216001.

Salimi-Khorshidi, G., Douaud, G., Beckmann, C.F., Glasser, M.F., Griffanti, L., Smith, S.M., 2014. Automatic denoising of functional mri data: combining independent component analysis and hierarchical fusion of classifiers. Neuroimage 90, 449–468. doi:10.1016/j.neuroimage.2013.11.046.

Schaefer, A., Kong, R., Gordon, E.M., Laumann, T.O., Zuo, X.N., Holmes, A.J., Eickhoff, S.B., Yeo, B.T., 2018. Local-global parcellation of the human cerebral cortex from intrinsic functional connectivity mri. Cerebral cortex 28, 3095–3114. doi:10.1093/cercor/bhx179.

Schomburg, I., Jäde, A., Ebeling, C., Gremse, M., Heldt, C., Huhn, G., Schomburg, D., 2004. Brenda, the enzyme database: Updates and major new developments. Nucleic acids research 32, D431–3. doi:10.1093/nar/gkh081.

Shervashidze, N., Schweitzer, P., van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M., 2011. Weisfeiler-lehman graph kernels. Journal of Machine Learning Research (JMLR) 12, 2539–2561.

Shlomi, J., Battaglia, P., Vlimant, J.R., 2020. Graph neural networks in particle physics. Machine Learning: Science and Technology 2, 021001. doi:10.1088/2632-2153/abbf9a.

Sillitoe, I., Bordin, N., Dawson, N., Waman, V.P., Ashford, P., Scholes, H.M., Pang, C.S.M., Woodridge, L., Rauer, C., Sen, N., Abbasian, M., Le Cornu, S., Lam, S.D., Berka, K., Varekova, I.H., Svobodova, R., Lees, J., Orengo, C.A., 2020. Cath: increased structural coverage of functional space. Nucleic Acids Research 49, D266–D273. doi:10.1093/nar/gkaa1079.

Simonovsky, M., Komodakis, N., 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs, in: Conference on Computer Vision and Pattern Recognition, IEEE Computer Society. pp. 29–38. doi:10.1109/CVPR.2017.11, arXiv:1704.02901.

Smith, S.M., Hyvärinen, A., Varoquaux, G., Miller, K.L., Beckmann, C.F., 2014. Group-pca for very large fmri datasets. Neuroimage 101, 738–749. doi:10.1016/j.neuroimage.2014.07.051.

Song, X., Frangi, A., Xiao, X., Cao, J., Wang, T., Lei, B., 2020. Integrating similarity awareness and adaptive calibration in graph convolution network to predict disease, in: Martel, A.L., Abolmaesumi, P., Stoyanov, D., Mateus, D., Zuluaga, M.A., Zhou, S.K., Racoceanu, D., Joskowicz, L. (Eds.), Medical Image Computing and Computer Assisted Intervention, Springer International Publishing, Cham. pp. 124–133. doi:10.1007/978-3-030-59728-3_13.

Sserwadda, A., Rekik, I., 2021. Topology-guided cyclic brain connectivity generation using geometric deep learning. Journal of Neuroscience Methods 353, 108988. doi:10.1016/j.jneumeth.2020.108988.

Uesawa, Y., 2016. Rigorous selection of random forest models for identifying compounds that activate toxicity-related pathways. Frontiers in Environmental Science 4. doi:10.3389/fenvs.2016.00009.

Verma, S., Zhang, Z.L., 2019. Deep universal graph embedding neural network. CoRR abs/1909.10086. arXiv:1909.10086.

Vivar, G., Kazi, A., Burwinkel, H., Zwergal, A., Navab, N., Ahmadi, S.A., 2021. Simultaneous imputation and classification using multigraph geometric matrix completion (mgmc): Application to neurodegenerative disease classification. Artificial Intelligence in Medicine

117, 102097. URL: https://www.sciencedirect.com/science/article/pii/S0933365721000907, doi:https://doi.org/10.1016/j.artmed.2021.102097.

Vivar, G., Zwergal, A., Navab, N., Ahmadi, S.A., 2018. Multi-modal disease classification in incomplete datasets using geometric matrix completion, in: Stoyanov, D., Taylor, Z., Ferrante, E., Dalca, A.V., Martel, A.L., Maier-Hein, L., Parisot, S., Sotiras, A., Papiez, B.W., Sabuncu, M.R., 0001, L.S. (Eds.), Graphs in Biomedical Image Analysis and Integrating Medical Imaging and Non-Imaging Modalities. Springer. volume 11044, pp. 24–31. doi:10.1007/978-3-030-00689-1_3, arXiv:1803.11550.

Von Luxburg, U., 2007. A tutorial on spectral clustering. Statistics and computing 17, 395–416. doi:10.1007/s11222-007-9033-z, arXiv:0711.0189.

Wale, N., Watson, I., Karypis, G., 2008. Comparison of descriptor spaces for chemical compound retrieval and classification. Knowledge and Information Systems 14, 347–375. doi:10.1109/ICDM.2006.39.

Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M., 2019. Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog) 38, 1–12. doi:10.1145/3326362.

Wu, D., Li, X., Feng, J., 2021. Multi-hops functional connectivity improves individual prediction of fusiform face activation via a graph neural network. Frontiers in Neuroscience 14. doi:10.3389/fnins.2020.596109.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S., 2019. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems 32, 4–24. doi:10.1109/tnnls.2020.2978386, arXiv:1901.00596.

Wu, Z., Ramsundar, B., Feinberg, E.N., Gomes, J., Geniesse, C., Pappu, A.S., Leswing, K., Pande, V., 2018. Moleculenet: A benchmark for molecular machine learning. Chemical Science 9, 513–530. doi:10.48550/ARXIV.1703.00564, arXiv:1703.00564.

Xing, X., Li, Q., Wei, H., Zhang, M., Zhan, Y., Zhou, X.S., Xue, Z., Shi, F., 2020. Ds-gcns: Connectome classification using dynamic spectral graph convolution networks with assistant task training. doi:10.48550/ARXIV.2001.03057, arXiv:2001.03057.

Xu, K., Hu, W., Leskovec, J., Jegelka, S., 2019. How powerful are graph neural networks?, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J., 2018. Hierarchical graph representation learning with differentiable pooling, in: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), Annual Conference on Neural Information Processing Systems, Curran Associates, Inc.. pp. 4805–4815.

Yun, S., Jeong, M., Kim, R., Kang, J., Kim, H.J., 2019. Graph transformer networks, in: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E.B., Garnett, R. (Eds.), Annual Conference on Neural Information Processing Systems, pp. 11960–11970.

Zaslavskiy, M., Jégou, S., Tramel, E.W., Wainrib, G., 2018. Toxicblend: Virtual screening of toxic compounds with ensemble predictors. Computational Toxicology 10, 81–88. doi:10.1016/j.comtox.2019.01.001, arXiv:1806.04449.

Zhan, K., Chang, X., Guan, J., Chen, L., Ma, Z., Yang, Y., 2019. Adaptive structure discovery for multimedia analysis using multiple features. IEEE Transactions on Cybernetics 49, 1826–1834. doi:10.1109/TCYB.2018.2815012.

Zhang, L., Wang, L., Zhu, D., 2020a. Recovering brain structural connectivity from functional connectivity via multi-gcn based generative adversarial network, in: Martel, A.L., Abolmaesumi, P., Stoyanov, D., Mateus, D., Zuluaga, M.A., Zhou, S.K., Racoceanu, D., Joskowicz, L. (Eds.), Medical Image Computing and Computer Assisted Intervention, Springer International Publishing, Cham. pp. 53–61. doi:10.1007/978-3-030-59728-3_6.

Zhang, M., Chen, Y., 2018. Link prediction based on graph neural networks, in: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), Annual Conference on Neural Information Processing Systems, pp. 5171–5181.

Zhang, M., Cui, Z., Neumann, M., Chen, Y., 2018. An end-to-end deep learning architecture for graph classification, in: McIlraith, S.A., Weinberger, K.Q. (Eds.), Proceedings of the AAAI conference on artificial intelligence, Association for the Advancement of Artificial Intelligence (AAAI). pp. 4438–4445. doi:10.1609/aaai.v32i1.11782.

Zhang, W., Zhan, L., Thompson, P.M., Wang, Y., 2020b. Deep representation learning for multimodal brain networks, in: Martel, A.L., Abolmaesumi, P., Stoyanov, D., Mateus, D., Zuluaga, M.A., Zhou, S.K., Racoceanu, D., Joskowicz, L. (Eds.), Medical Image Computing and Computer Assisted Intervention, Springer. pp. 613–624. doi:10.1007/978-3-030-59728-3_60.

Zhang, Z., Bu, J., Ester, M., Zhang, J., Yao, C., Yu, Z., Wang, C., 2019. Hierarchical graph pooling with structure learning abs/1911.05954. doi:10.48550/ARXIV.1911.05954, arXiv:1911.05954.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M., 2020. Graph neural networks: A review of methods and applications. AI Open 1, 57–81. doi:10.1016/j.aiopen.2021.01.001.

Zhu, Y., Ma, J., Yuan, C., Zhu, X., 2022. Interpretable learning based dynamic graph convolutional networks for alzheimer's disease analysis. Inf. Fusion 77, 53–61. doi:10.1016/j.inffus.2021.07.013.

Zitnik, M., Agrawal, M., Leskovec, J., 2018. Modeling polypharmacy side effects with graph convolutional networks. Bioinformatics 34, i457–i466. doi:10.1093/bioinformatics/bty294.

Zitnik, M., Nguyen, F., Wang, B., Leskovec, J., Goldenberg, A., Hoffman, M.M., 2019. Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities. Information Fusion 50, 71–91. doi:10.1016/j.inffus.2018.09.012, arXiv:1807.00123.

Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. Journal of the royal statistical society: series B (statistical methodology) 67, 301–320. doi:10.1111/j.1467-9868.2005.00503.x.