

# New Graph-based Representation Learning Algorithms

by

**Yanfu Zhang**

Master of Science, University of Rochester, 2017

Submitted to the Graduate Faculty of  
the Swanson School of Engineering in partial fulfillment  
of the requirements for the degree of  
**Doctor of Philosophy**

University of Pittsburgh

2023

UNIVERSITY OF PITTSBURGH  
SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Yanfu Zhang

It was defended on

June 27th 2023

and approved by

Wei Gao, PhD, Associate Professor, Department of Electrical and Computer Engineering

Liang Zhan, PhD, Associate Professor, Department of Electrical and Computer Engineering

Zhi-Hong Mao, PhD, Professor, Department of Electrical and Computer Engineering

Wei Chen, PhD, Professor, Department of Pediatrics, School of Medicine

Dissertation Director: Heng Huang, PhD, John A. Jurenko Endowed Professor, Department  
of Electrical and Computer Engineering

Copyright © by Yanfu Zhang  
2023

# New Graph-based Representation Learning Algorithms

Yanfu Zhang, PhD

University of Pittsburgh, 2023

In recent years, graph neural networks (GNN) have succeeded in many structural data analyses, including information retrieval, recommendation system, and social network analysis. Although the first-order graph convolutional networks are initially designed for single-view node-level representation learning, the graph-level analysis using GNNs applies according to recent studies, e.g., the shared structures can be learned from single-view graph data. The dissertation focuses on the efficient and robust graph-level representation learning using GNNs. Several effective methods are proposed to address the critical problems in graph representation learning, including over-smoothing, graph structural difference, and fast training. The application of graph representation learning on medical data is also studied, including new methods for single-view, multi-view, and unsupervised medical data analysis.

# Table of Contents

<b>Preface</b> . . . . .	xiv
<b>1.0 Introduction</b> . . . . .	1
<b>2.0 Improving Network Embedding via New Second-Order Continuous Graph Neural Networks</b> . . . . .	3
2.1 Motivation . . . . .	3
2.2 Related Work . . . . .	5
2.3 Proposed Method . . . . .	7
2.3.1 Second-Order Continuous GNNs . . . . .	7
2.3.1.1 Revisiting Messaging Passing Neural Networks and First-Order Continuous GNNs . . . . .	7
2.3.1.2 Second-Order Continuous Graph Neural Networks . . . . .	9
2.3.2 Implementation of Second-Order Continuous GNN . . . . .	10
2.3.3 Enhanced Explanation for Graph Neural Networks Using Second-Order Information . . . . .	10
2.3.3.1 Problem Formulation . . . . .	11
2.3.3.2 Semi-model-aware GNN Explainer . . . . .	12
2.3.4 First-Order v.s. Second-Order . . . . .	13
2.4 Experimental Results . . . . .	16
2.4.1 Datasets and Experiment Settings . . . . .	16
2.4.2 Baseline Methods . . . . .	18
2.4.3 Comparison Results . . . . .	18
2.4.3.1 Semi-Supervised Node Classification . . . . .	18
2.4.3.2 Graph Classification . . . . .	19
2.4.3.3 Prediction Explanation . . . . .	19
2.4.4 Analysis of Model Parameters . . . . .	20
2.4.4.1 The model performance v.s. Integration Time . . . . .	20

2.4.4.2	The Effect of Damping Term . . . . .	20
2.5	Conclusion . . . . .	20
<b>3.0</b>	<b>Robust Self-Supervised Structural Graph Neural Network . . . . .</b>	<b>26</b>
3.1	Motivation . . . . .	26
3.1.1	Notations . . . . .	28
3.2	Related Works . . . . .	29
3.2.1	Graph Representation Learning . . . . .	29
3.2.2	Graph Contrastive Learning . . . . .	29
3.2.3	Network Proximity . . . . .	30
3.2.4	Deep Implicit Layers . . . . .	31
3.3	Proposed Method . . . . .	31
3.3.1	Self-Supervised Graph Neural Network via Wasserstein Proximity . . . . .	33
3.3.2	(Asymptotically) Distributionally Robust Contrastive Learning . . . . .	35
3.3.2.1	Distributionally Robust Contrastive Learning . . . . .	35
3.3.2.2	Asymptotically Distributionally Robust Contrastive Learning . . . . .	36
3.3.3	Computing Wasserstein Distance via Deep Implicit Layer . . . . .	37
3.3.4	Algorithm and Implementation . . . . .	39
3.4	Experimental Results . . . . .	40
3.4.1	Training Self-Supervised Graph Neural Network . . . . .	41
3.4.2	Downstream Analysis . . . . .	41
3.4.2.1	Node Classification . . . . .	41
3.4.2.2	Graph Classification . . . . .	42
3.4.2.3	Top-k Similarity Search . . . . .	42
3.4.3	Ablation Studies . . . . .	43
3.4.3.1	Model Robustness . . . . .	43
3.4.3.2	Cardinality of Embedding Set . . . . .	43
3.4.3.3	Computational Time . . . . .	44
3.5	Conclusion . . . . .	44
<b>4.0</b>	<b>Training Graph Neural Network Faster using Stale Gradients . . . . .</b>	<b>51</b>
4.1	Motivation . . . . .	51

4.1.1	Organization and Notations . . . . .	52
4.2	Preliminary and Related Work . . . . .	54
4.2.1	Preliminary . . . . .	54
4.2.2	Related Work . . . . .	55
4.3	Gradient Flashback Method . . . . .	56
4.4	Convergence Analysis . . . . .	60
4.5	Appendix . . . . .	62
4.5.1	Proof of Theorem 1 . . . . .	62
4.5.2	Proof of Theorem 2 . . . . .	65
4.6	Experiments . . . . .	66
4.6.1	Experiment Settings . . . . .	66
4.6.2	Comparison with Baselines . . . . .	66
4.6.3	Convergence Results . . . . .	67
4.6.4	Efficiency Under Various Settings . . . . .	68
4.7	Conclusion . . . . .	69
<b>5.0</b>	<b>Applying Graph Representation Learning to Varied Medical Imaging</b>	
	<b>Problems . . . . .</b>	<b>74</b>
5.1	Learning Shared Structure from Single-view Graph data . . . . .	74
5.1.1	Motivation . . . . .	74
5.1.2	Methodology . . . . .	75
5.1.2.1	Preliminary . . . . .	76
5.1.2.2	Graph Convolutional Network without Pre-defined Graph Structure . . . . .	77
5.1.2.3	Can we apply GCN to graph data without predefined graph structure? . . . . .	77
5.1.2.4	How can we find a “good” graph structure for brain connectome? . . . . .	79
5.1.2.5	Is GCN preferred compared to naive neural network? . . . . .	82
5.1.3	Pseudo-Label . . . . .	82
5.1.4	Experiments . . . . .	84
5.1.4.1	Data Description . . . . .	84

5.1.4.2	Experiment Setting . . . . .	85
5.1.4.3	Performance Comparison . . . . .	86
5.1.5	Conclusion . . . . .	88
5.2	Heterogeneous Graph Neural Networks Integrating Multi-view graphs . . . . .	88
5.2.1	Motivation . . . . .	88
5.2.2	Methodology . . . . .	89
5.2.2.1	Preliminary . . . . .	89
5.2.2.2	Predicting PD Clinical Scores via Heterogeneous GCN . . . . .	90
5.2.3	Experiments . . . . .	93
5.2.3.1	Data Description . . . . .	93
5.2.3.2	Experiment Settings . . . . .	94
5.2.3.3	Results . . . . .	95
5.2.4	Conclusion . . . . .	95
5.3	Unsupervised Multi-View Graph Representation Learning . . . . .	97
5.3.1	Motivation . . . . .	97
5.3.2	Methodology . . . . .	99
5.3.3	Experimental Results . . . . .	103
5.3.4	Conclusion . . . . .	107
<b>6.0</b>	<b>Conclusion . . . . .</b>	<b>108</b>
	<b>Bibliography . . . . .</b>	<b>110</b>



## List of Tables

1	Statistics of datasets for node classification. . . . .	22
2	Statistics of datasets for graph classification. . . . .	22
3	Node classification results on citation networks. The Cora, Citeseer, and Pubmed columns summarize the results using standard test-training split. The corresponding columns with superscript * summarize the results over 10 random splits test-training split. The values for GCN-GODE and GAT-GODE are taken from the original paper. Best-performing methods are bold faced, and the runner-ups are underlined. For the random split, The results of GCN-GODE and GAT-GODE are not available and are denoted by $-$ . . . . .	23
4	Graph Classification results. . . . .	24
5	Explanation AUC results. . . . .	24
6	The statistic for the pre-training datasets. Among these datasets, Academia, DBLP-SNAP, and DBLP-NetRep are academic datasets, and the rest are social networks. . . . .	45
7	Node Classification Results. . . . .	46
8	Graph Classification Results. . . . .	47
9	Top-k Similarity Search ( $k = 40$ ). Best-performing self-supervised methods are bold faced. Best performing non-self-supervised methods are denoted by *. . . . .	48
10	The comparison of per-step time for the baseline and our method using different subgraph size (denoted by superscripts). . . . .	50
11	Dataset description and model/training details. . . . .	71
12	Comparison of different training methods on multiple tasks and multiple datasets. Top: running time of different methods, the best results are marked in bold font; bottom: test accuracy of the models trained using different methods. The time efficiency and consistent performance can be observed from the values. . . . .	71

13	Quantitative comparison of baselines and the proposed method. Both the mean and the standard error are given, and the best results are bold faced. Metrics without significant difference between baselines and FCGCN are denoted with $\star$ ; the metric without significant difference between FCGCN and kGCN/RGCN is denoted with $\diamond$ . . . . .	86
14	The comparison of the proposed method with baselines. For both metrics, smaller values indicate better results. The values are displayed as $\text{mean}(\mu) \pm \text{standard deviation} (\sigma)$ from five tests. Bold font indicates the best performance. . . . .	96
15	Predictions with different combination of modals. Values follow the instruction in Table. 14. Bold font indicates the best performance. . . . .	96
16	The comparison on classification tasks. . . . .	105
17	The comparison on regression tasks. . . . .	106

## List of Figures

1	Learn the second order GNN. . . . .	7
2	A ball-spring system under the second order dynamics (elastic force, gravity, and friction). . . . .	21
3	The pipeline for subgraph sampling. We compute the edge embeddings using a deep neural network. We use the invertible Gaussian reparameterization trick to sample the subgraph, which allows the straight-through gradients estimation for the back-propagation. We update the embedding network using Monte Carlo method to optimize the mutual information between the explanation and the computation graph. . . . .	21
4	Training loss (left) and validation accuracy (right) under different integration time.	25
5	Training loss (left) and validation accuracy (right) with different damping terms.	25
6	Subgraphs generated from the same node (colored in red) are similar, and from different nodes are dissimilar (red <i>v.s.</i> blue). To learn a robust encoder, we fix the keys, <i>i.e.</i> $G_{k_i}$ and $G_{k_j}$ , and focus on the most difficult query $G_{q_i}$ (solid red line) instead of random queries (dashed red lines). . . . .	32
7	The $r$ -ego subgraphs surrounding the nodes of interest are fed into the structural GNN to obtain the embedding sets. Wasserstein distance is then computed as the network proximity. . . . .	34
8	An example shows the distribution of $r$ -ego subgraphs are determined by the sampling methods. The node of interest is colored in red, and three possible subgraphs are shaded in different colors (middle column). Under different sampling techniques ( <i>e.g.</i> altering the backward jump probability in random walks), the distribution of subgraphs are presumably different (right column). . . . .	45

9	The model performance under different subgraph distributions. We test five different sampling settings for the second stage: for neighborhood sampling with different neighbor size, we consider NS 4 and NS 5; for random walk with restart with different restart probability, we consider RWR 0.6, RWR 0.7 and RWR 0.8. The results are based on 10-fold validation accuracy on RDT-B. Our model performs similar under different settings, which indicates our method is robust to distribution shift. . . . .	49
10	Model performance <i>v.s.</i> sampling size. . . . .	50
11	The growing of the receptive fields layer-wise. Top line: the growing for exact method; bottom line: the growing for sampled methods. Involved nodes in each nodes are labelled with different colors. Clearly, deep network causes the size of involved nodes in bottom layers exploded, particularly for exact method. . . . .	53
12	The training process of the proposed method. Naive GCN model are splitted into multiple sequential sub-models. In the forward step, the outputs are computed using current model; in the backward step, the updating is computed using stale error gradients. We name it <i>gradient flashback</i> due to the re-occurrence of historical gradients. . . . .	70
13	The results of loss against time on training set. For all datasets, the proposed method converges much faster than naive methods, given the same sampling scheme. In the long run, it also attains an identical performance with naive stochastic methods, which verifies the convergence analysis. . . . .	72
14	The results of accuracy against epoch on test set. For all datasets, the proposed method attains an identical performance with naive stochastic methods, given the same sampling scheme. . . . .	72
15	Running time per epoch. The proposed method consumes much less time compared to naive stochastic methods using the same sampling scheme. . . . .	73
16	The iterative procedure of random graph generation, GCN training, and embeddings inference. . . . .	79
17	The structure of the proposed method. Brain connectome is transformed into embeddings using GCN and MLP is used for regression successively. . . . .	81

18	Comparison of training curves of FCGCN and RGCN. . . . .	85
19	Visualization of clustering of clinical depression scores using t-SNE and PCA respectively, from a random start. The clustering is obtained through K-Means with $K = 4$ and 8 . . . . .	87
20	The proposed heterogeneous GCN for PD clinical scores prediction. (a) illustrates the entire structure, in which multi-modal brain networks are generated from MRIs, and sequentially processed by GCN and MLP; (b) depicts the stacked convolutional layer and pooling layer; (c) provides a detailed description of the pooling procedure, including node merge, graph distillation and feature pooling. . . . .	91
21	The structure of the proposed method. Each view uses an independent VGAE to learn a unified $\mu$ , while the $\sigma$ is different. . . . .	98
22	Left: a simple DNN. Right: the corresponding DAG. Each edge represents a network, and each node denote an intermediate representation. . . . .	100
	a    Round-Robin . . . . .	102
	b    Proportionality . . . . .	102
24	Left to right: ADNI, NACC, PPMI. . . . .	107

## Preface

I would like to extend my deepest gratitude to Dr. Heng Huang for his unwavering support and guidance as my advisor throughout these past several years. His expertise and mentorship have been invaluable in shaping and conducting this research. I am immensely grateful for the publication opportunities, travel assistance, and research support he has provided during this journey.

I would also like to express my sincere appreciation to Professor Wei Chen, Professor Wei Gao, Professor Zhihong Mao, and Professor Liang Zhan for their gracious commitment to serving on my PhD dissertation committee. Their insightful suggestions and constructive feedback on the dissertation study and future research plans have been instrumental in its development. I am truly grateful for the time and guidance they have dedicated to my academic growth.

I am also indebted to every collaborator who has contributed to my PhD journey. I would like to extend a special thanks to Dr. Liang Zhan and Dr. Wei Chen for their invaluable guidance and support in my research endeavors. Additionally, I want to express my gratitude to Professor Feihu Huang, Professor Lei Luo, Professor Feng Zheng, Professor Xiaoqian Wang, Professor Hongchang Gao, Dr. Zhouyuan Huo, Guodong Liu, Shangqian Gao, Wenhan Xian, Dr. Runxue Bao, An Xu, Dr. Haoteng Tang, Xiaotian Dou, Junyi Li, Zhengmian Hu, Xidong Wu, and Yihan Wu for their collaboration and support. It has been an honor and privilege to work alongside such talented individuals. I am grateful for the opportunity to learn from them and for the valuable friendships we have formed.

Lastly, I would like to express my heartfelt appreciation to my parents for their unwavering love and boundless support throughout my academic journey. Their encouragement and belief in my abilities have been the driving force behind my achievements. I am eternally grateful for everything they have done for me, and I recognize that this milestone would not have been possible without their unwavering dedication.

## 1.0 Introduction

In recent years, graph neural networks (GNN) have achieved significant success in various structural data analyses, such as information retrieval, recommendation systems, and social network analysis. Graph neural networks can learn representations at both the node-level and graph-level, which can be utilized in a wide range of downstream analyses. This proposal aims to study several crucial problems in graph representation learning systematically and introduces new models and optimization methods.

Our contributions encompass both methodology and application. From a methodology standpoint, we propose a framework that utilizes second-order graph neural networks, which typically exhibit a more flexible transformation than their first-order counterparts. Additionally, we introduce a semi-model-agnostic method based on our model to enhance prediction explanations by incorporating high-order information. We learn structural embeddings, where proximity is characterized using the 1-Wasserstein distance. To achieve this, we put forward a distributionally robust self-supervised graph neural network framework for learning these representations. The embeddings are computed based on subgraphs centered around the node of interest, representing the node itself and its neighbors. Moreover, our model is end-to-end trainable. Furthermore, we propose a faster training method for GCN. The bottleneck in training GCN arises from the network depth and the sampling behavior. Rather than improving the sampling process, we focus on 'reducing' the depth by reusing stale gradients.

In terms of applications, we demonstrate the effectiveness of graph representation learning in medical data analysis. We employ GNNs to learn the shared graph structure, characterizing the graph distribution based on a small-world model. Specifically, we show that GCN can be applied even without a given graph structure by utilizing a naive complete graph. Furthermore, we propose a method to learn the graph structure from data, enhancing the performance of GCN by generating random graphs with small-world properties for model training. We introduce a heterogeneous GCN for multi-view networks. We propose an adaptive pooling scheme driven by graph structure and network patterns. This scheme benefits from gathering local information, resulting in a compact yet informative graph and providing computational

and training efficiency. Moreover, we propose to learn unified representations from multi-modal brain networks using unsupervised learning techniques. To enhance the generalization ability of the learned representations for different downstream analyses, these representations should exhibit disentanglement and proportionality for different modalities. Our approach ensures that the learned representations effectively capture information from various modalities and can be leveraged by a wide range of downstream analyses.



## 2.0 Improving Network Embedding via New Second-Order Continuous Graph Neural Networks

### 2.1 Motivation

Recently, there is growing interest in designing graph neural networks (GNN) to solve a variety of web research problems, such as social network analysis [91], recommendation systems [143], fraud detection [183], natural languages [79], brain connectome analysis [177], *etc.* Graph neural networks are primitively motivated by graph convolution [30, 64] based on spectral graph theory. Consecutive researches [38, 149, 47, 153, 50, 154, 173] show GNNs are powerful in learning graph embeddings, and the simplicity and effectiveness of GNN can be boosted via generalizing the graph convolution operation to aggregate the information from the nodes of interest and their neighbors. Some representative variants include GraphSage [50], Message Passing Neural Network [153], Graph Attention Network [130] and Graph Isomorphism Network [149].

There is some discrepancy between the practical requirements for business scenarios and the existing research. For example, one usually wants to make a personalized recommendation targeting on individual users. Deeper networks usually are beneficial for learning more complex data distributions. But node representations may be over-smoothed [64, 96, 149] due to the low-frequency filtering property of many GNNs. As a result, the user difference is blurred, which becomes a hindrance if one wants to make personalized recommendation. Another example is that it is difficult to build business decision based on the black-box predictions of GNNs [152, 81, 159]. In social networks, finding influential neighbors can significantly improve the understanding of the user behavior. As such, there is some interest in explaining the GNN predictions based on a small subgraph.

To address these problems, in this paper, we propose a novel second-order continuous graph neural network. There were some attempts to address over-smoothing problem [149, 76]. Inspired by an emerging research topic connecting the residual network and ordinary differential equations [25], continuous-depth GNNs was proposed based on the dynamics

of hidden layers and show advantages than the discretized counterparts in better memory efficiency and superior performance. The first-order frameworks adopted by existing graph ODEs characterize the diffusion process which cannot fully address the over-smoothing problem because the heat kernel is usually the Gaussian. Besides, many methods [103, 146] use augmented features as an ad-hoc step for numerical stability. The selection of feature augmentation is thus discretionary and lacks interpretability. Our approach learns the second-order dynamics of GNNs and considers the effect of a force term and a damping term to avoid over-smoothing, which can be viewed as discretizing the variants of wave equations. Our model can also be viewed as a coupled first-order equation with interpretable augmented feature velocity. Previous GNN explanation methods are usually model-agnostic, which omits the second-order information in our model. We design a new semi-model-agnostic method that explicitly considers the high-order information but leaves the GNN structure untouched to enhance the prediction explanation.

Our contributions can be summarized as follows:

- We propose a second-order continuous GNN based on the message passing neural network framework. Our second-order model can be expressed as a coupled first-order equation via augmenting the node features with their gradients, and the implementation is feasible using the popular first-order framework.
- We construct an analog between continuous GNNs and partial differential equations because graph Laplacian can be viewed as discretized Laplacian operator on manifolds.
- We introduce a method to explain the GNN prediction via leveraging the high-order information with an independent edge embedding network. The explanation is agnostic to the specific GNN structure.
- We conduct extensive experiments on several graphs datasets of homophily, which is an important property of social networks [86, 100, 186]. The results show that our method outperforms related baselines in various downstream tasks, and the second-order information can boost the model explanation.

## 2.2 Related Work

**Neural ODEs:** Residual network [53] is an important method to push neural networks to extreme depths. It is observed that the skip links in ResNet can be seen as an Euler discretization of a continuous transformation. Based on this connection, the research on neural ODEs evokes emerging interest. A popular framework, Neural ODEs [25], considers the continuous-depth neural networks via taking the limit of this discretization. The optimization *w.r.t.* the ODE solvers adopt the adjoint sensitivity method [104], which treats the ODE solve as a black box and solve a second augmented ODE backward. The continuous depths make neural ODEs suitable for modeling the dynamics of complex systems, particularly those that cannot be described analytically. Meanwhile, it is difficult for standard neural networks to learn symmetries and conservation laws while neural ODEs can address this issue. For example, Hamiltonian Neural Networks [48] and Lagrangian Neural Networks [29] can produce energy-conserving models for various tasks.

**Augmented Neural ODEs** In Neural ODEs, the hidden states evolves continuously according to a differential equation, whose velocity is described by a neural network. Once the dynamics are known, the gradients of some objective with regard to the model parameters can be computed through adjoint sensitivity method. It is shown that the feature mapping learned by Neural ODEs is a homeomorphism [36], which implies the existence of functions Neural ODEs cannot represent. Augmented Neural ODEs [36] is proposed to address these limitations via introducing additional features. A related variant of augmented neural ODEs is the Second Order Neural ODEs [95]. SONODE explicitly considers the velocities as augmented dimensions, and proposes to learn the acceleration. The inductive bias introduced in SONODE is beneficial in learning many dynamical systems governed by second order laws, such as Newton’s equations of motion and oscillators. SONODE can be solved either using the adjoint state directly or using the coupled first order ODE. Of note, augmented neural ODEs can also learn high-order dynamics. However, the high order dynamics learned by ANODE are characterized by abstract functions and the augmented features typically are entangled, which leads to difficulties in interpretability.

**Graph Representation Learning:** Graph representation learning is featured by mining

the topological structures of graphs and encoding nodes with low-dimensional embeddings. Representative works collect local patterns and learn mappings from graphs to vectors, including Word2Vec [87], DeepWalk [102], and LINE [124]. To simultaneously exploit the structural knowledge and the enriched side information in attribute graphs, increasing interests are paid to GNNs. GNN [64] is based on graph convolution, an extension of regular convolutions that can process structural data. Via designing a more efficient aggregation mechanism, GraphSage [153, 50, 154] and Message Passing Neural Network (MPNN) [47] broaden the application of GNNs to the analysis for large-scale graphs. Graph Attention Network (GAN) [130] introduces an attention mechanism into general network analysis, and self-attention is proposed for graph pooling [74] for graph classification tasks. Based on the relationship between GNNs and graph isomorphism problem, Graph Isomorphism Network (GIN) [149] and Deep Graph Convolutional Neural Network(DGCNN) [166] are designed. On the other hand, there is some attempt to learn interpretable graph representation [152], particularly for some applications, e.g., brain networks [122, 123].

**Continuous-depth Graph Neural Networks** The idea behind continuous-depth neural networks is also employed in graph neural networks to handle structural data. Graph neural ordinary Differential Equations (GDEs) [103] propose a continuum of GNN layers to characterize the input-output relationship. In detail, the static models and Spatio-temporal models are developed to handle different tasks. Alternatively, Continuous Graph Neural Networks (CGNN) [146] proposes a propagation scheme inspired by diffusion-based methods. Specifically, the representations learned by CGNN have entangled *w.r.t.* the graph structure and the node features while the terminal time goes to infinite time. The continuous-depth formulation of GNNs also inspires some related methods. For example, the numerical gap between the discrete and continuous graph diffusion process can affect the model performance [141], and Simple Graph Convolution [142] can be boosted by decoupling the terminal time and the finite difference steps. Continuous Graph Flow (CGF) [32], as a graph generative model, generalizes the messaging passing mechanism to continuous time.

## 2.3 Proposed Method

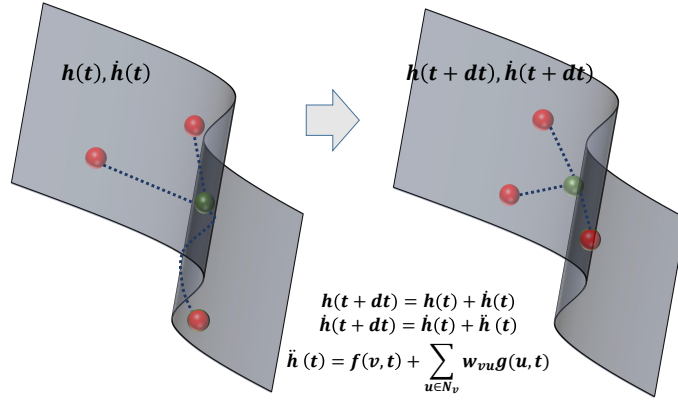


Figure 1: Learn the second order GNN.

In this section, we first review the Messaging Passing Neural Network (MPNN) and the first-order continuous GNN, then describe our second-order continuous GNNs. Figure. 6 illustrates the second-order dynamics of the node embeddings on a manifold, which will be detailed in the following. Via reformulating our model as coupled first-order neural ODEs, we give a simple implementation. We propose a method to explain the GNN prediction jointly using the first and the second order information. At last, motivated by the connection between graph Laplacian and the Laplace-Beltrami operator, we discuss some properties of continuous GNNs from the view of neural partial differential equations instead of ordinary differential equations.

### 2.3.1 Second-Order Continuous GNNs

#### 2.3.1.1 Revisiting Messaging Passing Neural Networks and First-Order Continuous GNNs

Existing continuous GNNs are based on the continuous counterpart of residual structures in GNNs. In this part, we first describe the Messaging Passing Neural Network (MPNN), then recap the first-order continuous GNN under this framework.

We denote a graph  $\mathcal{G}$  by its node set  $\mathcal{V}$  and its edge set  $\mathcal{E}$ . For a single node  $v \in \mathcal{V}$ , we use  $\mathcal{N}(v) := \{u \in \mathcal{V} : (v, u) \in \mathcal{E} \vee (u, v) \in \mathcal{E}\}$  to denote its neighbors. A MPNN layer performs a spatial-based convolution on  $v$ , and the representation of  $v$  at layer  $l + 1$  is defined on the representations of  $v$  and its neighbor  $\mathcal{N}(v)$  at layer  $l$ ,

$$\mathbf{h}^{(v)}(l + 1) = \mathbf{u} \left[ \mathbf{h}^{(v)}(l), \sum_{u \in \mathcal{N}(v)} \mathbf{m}(\mathbf{h}^{(v)}(l), \mathbf{h}^{(u)}(l)) \right]. \quad (2.1)$$

Specifically, we have  $\mathbf{h}^{(v)}(0) = \mathbf{x}_v$ , which is the input to the MPNN.  $\mathbf{x}_v$  can be node features or some representations learned by an encoder.  $\mathbf{u}$  and  $\mathbf{m}$  are functions with trainable parameters.

A general first-order continuous GNN can be defined by first setting  $\mathbf{u}(\mathbf{x}, \mathbf{y}) := \mathbf{x} + \mathbf{g}(\mathbf{x}, \mathbf{y})$  [103], where  $\mathbf{g}(\mathbf{h}^{(v)}(l), \mathbf{h}^{(\mathcal{N}(v))}(l)) = \mathbf{g}\left(\sum_{u \in \mathcal{N}(v)} \mathbf{m}(\mathbf{h}^{(v)}(l), \mathbf{h}^{(u)}(l))\right)$  is some function, (2.1) can be written as,

$$\mathbf{h}^{(v)}(l + 1) - \mathbf{h}^{(v)}(l) = \mathbf{g}(\mathbf{h}^{(v)}(l), \mathbf{h}^{(\mathcal{N}(v))}(l)). \quad (2.2)$$

By interpreting the layer  $l$  as a discretization step in time  $t$ , the continuous-depth counterpart of MPNN is defined on the continuity equation of the above difference equation,

$$\dot{\mathbf{h}}^{(v)}(t) = \mathbf{f}_{MPNN}^{(v)}(\mathbf{H}, \boldsymbol{\theta}) := \mathbf{g}(\mathbf{h}^{(v)}(t), \mathbf{h}^{(\mathcal{N}(v))}(t)). \quad (2.3)$$

Here  $\mathbf{H}$  refers to all node features, and  $\boldsymbol{\theta}$  refers to the model parameters, which defines the differential function. Formally, given input node features, the continuous MPNNs compute the node representations by solving the Cauchy problem,

$$\dot{\mathbf{h}}^{(v)}(t) = \mathbf{g}(\mathbf{h}^{(v)}(t), \mathbf{h}^{(\mathcal{N}(v))}(t)), \quad \mathbf{h}^{(v)}(0) = \mathbf{x}_0^{(v)}. \quad (2.4)$$

Similar to neural ODEs, the forward pass is to solve the problem numerically using ODE solvers, and the optimization of parameters uses the adjoint sensitivity method.

### 2.3.1.2 Second-Order Continuous Graph Neural Networks

In this paper, we consider a second-order dynamics for GCN. Specifically, we alter the definition of  $\mathbf{u}$  and the second-order model is defined as,

$$\ddot{\mathbf{h}}^{(v)}(t) = \mathbf{f}_{MPNN}^{(v)}(\mathbf{H}, \boldsymbol{\theta}) := \mathbf{g}(\mathbf{h}^{(v)}(t), \mathbf{h}^{(\mathcal{N}(v))}(t)). \quad (2.5)$$

$\mathbf{g}(\cdot)$  considers the interaction the neighbors have on the center node  $v$ . More generally, we consider two additional terms for (2.5), a term for the velocity decay and a term only related to the center node. Finally, our second-order continuous GNNs solves the following Cauchy problem,

$$\ddot{\mathbf{h}}^{(v)}(t) + \alpha \dot{\mathbf{h}}^{(v)}(t) = \mathbf{f}(\mathbf{h}^{(v)}(t)) + \mathbf{g}(\mathbf{h}^{(v)}(t), \mathbf{h}^{(\mathcal{N}(v))}(t)), \quad (2.6)$$

$$\mathbf{h}^{(v)}(0) = \mathbf{x}_0^{(v)}, \quad \dot{\mathbf{h}}^{(v)}(0) = 0. \quad (2.7)$$

Here  $\alpha > 0$  is a small positive number,  $\mathbf{f}(\mathbf{h}^{(v)}(t))$  is the force term. The initial state for  $\dot{\mathbf{h}}^{(v)}(0)$  is set to 0 for simplicity. The differential and the node representations at time  $t$  is then given by  $\dot{\mathbf{h}}^{(v)}(t) = \int_0^t \ddot{\mathbf{h}}^{(v)}(t) dt$  and  $\mathbf{h}^{(v)}(t) = \mathbf{x}_0^{(v)} + \int_0^t \dot{\mathbf{h}}^{(v)}(t) dt$ .  $\alpha \dot{\mathbf{h}}^{(v)}(t)$  is usually called a damping term.

Our model has a straightforward physical explanation. Let every node in a graph represent a ball. The features describe the spatial position of the balls. The edges denote the existence of the interaction between balls, for example, we can think that two linked nodes are two balls connected by a spring. The damping term can be viewed as friction. The force term is associate with the medium a ball lies in, which is not dependent on the other balls. For example, the balls are placed on a curved surface and are under the influence of a component of weight. An one-dimensional example is illustrated in Fig. 2.

### 2.3.2 Implementation of Second-Order Continuous GNN

The forward pass of our model can be accomplished via off-the-shelf ODE solvers. Similar to the first-order scenario, the backpropagation can be performed via the adjoint sensitivity method [104], which treats ODE solvers as black-boxes and has a low memory cost. The adjoint state of (2.8) can be computed using Lagrangian methods.

**Proposition 1.** *The adjoint state of (2.8) follows the second order ODE  $\ddot{r} = r^\top \frac{\partial \ddot{\mathbf{h}}}{\partial \mathbf{h}^{(v)}} - \alpha \dot{r}^\top$ . The update of the model parameters is the integral  $\frac{dL}{d\theta} = - \int_t^0 r^\top \frac{\partial x}{\partial \theta} ds$ .*

Our model is a special case of Proposition 3.1 in [95], and the above result is obtained by substituting the damping  $\ddot{\mathbf{h}}$ . The second-order differential equations in our approach can also be viewed as first-order coupled differential equations [95]. This relation gives an alternative model of our second-order CGNN as first-order augmented Neural ODEs. More specifically, we augment the node features to include their differentials  $\mathbf{z}^{(v)} = [\mathbf{h}^{(v)}(t), \dot{\mathbf{h}}^{(v)}(t)]$ , then our second-order continuous GNN can then be represented as a first-order Cauchy problem,

$$\mathbf{z}^{(v)} = [\mathbf{h}^{(v)}(t), \dot{\mathbf{h}}^{(v)}(t)], \dot{\mathbf{z}}^{(v)} = [\dot{\mathbf{h}}^{(v)}(t), \ddot{\mathbf{h}}^{(v)}(t)], \mathbf{z}_0^{(v)} = [\mathbf{x}_0^{(v)}, 0]. \quad (2.8)$$

As pointed out in [95], the adjoint state of the second-order model is equivalent to the adjoint state of the coupled first-order model. The disentangled representation in second-order models may involve more computation than the entangled augmented NODE. Due to this reason, we use the first-order optimization for simplicity in our implementation. In detail, we maintain  $\mathbf{z}^{(v)}$  in our continuous GNNs, and in the forward pass  $\dot{\mathbf{z}}^{(v)}$  is obtained by concatenating  $\dot{\mathbf{h}}^{(v)}$  from the stale  $\mathbf{z}$  and the newly computed  $\ddot{\mathbf{h}}^{(v)}$ . No modification is required for the backward propagation.

### 2.3.3 Enhanced Explanation for Graph Neural Networks Using Second-Order Information

In the representation learning for graph data, the high-order part of  $\mathbf{z}$  is discarded. However, it can provide some information in explaining the prediction of GNNs, which indicates that our second-order continuous GNN enjoys better interpretability compared



to first-order methods. More specifically, in this section we propose a new method for example-level explanations via employing the first-order and the second-order information jointly.

### 2.3.3.1 Problem Formulation

We employ the definition of example-level explanation from [152]. Assume we have a trained GNN  $\hat{y} = f(\mathcal{G}_c, x_c)$ . Here  $\mathcal{G}_c(v)$  is a computation graph spanned from  $v \in \mathcal{G}$ , which is the induced subgraph on the full  $\mathcal{G}$  involved in computing the prediction for  $v$ . The associated adjacency matrix is  $A_c(v)$ .  $x_c(v)$  is the associated feature set  $\{x_j | v_j \in \mathcal{G}_c(v)\}$ . Using the computation graph instead of the full graph can greatly reduce the computational burden. Given a node  $v$ , its explanation is  $(\mathcal{G}_s, x_s(v))$ . Here  $\mathcal{G}_s$  is a small subgraph on the computation graph  $\mathcal{G}_c(v)$  and  $x_s(v)$  is a small subset of node features, masked out by  $\{x_j | v_j \in \mathcal{G}_s(v)\}$ .  $(\mathcal{G}_s, x_s(v))$  are important the prediction  $\hat{y}_v$ . The importance is evaluated using mutual information  $I$ ,

$$\max_{\mathcal{G}_s} I(y, (\mathcal{G}_s, x_s)) = H(Y) - H(Y | \mathcal{G} = \mathcal{G}_s, x = x_s), \quad (2.9)$$

here  $H$  is the entropy. Typical explanation-generation methods are model-agnostic by fixing the GNN  $f(\mathcal{G}_c, x_c)$ , therefore,  $H(Y)$  is constant. The second term can be bounded,

$$H(Y | \mathcal{G} = \mathcal{G}_s, x = x_s) = -\mathbb{E}_{Y | \mathcal{G}_s, x_s} [\log P_f(Y | \mathcal{G} = \mathcal{G}_s, X = X_s)], \quad (2.10)$$

We use  $A_s(v)$  to represent the adjacency matrix of  $\mathcal{G}_s$ . Of note, model-agnostic methods fail to make use the second-order information in our model. To address this problem, we provide a semi-model-aware explanation-generation method in the following.

### 2.3.3.2 Semi-model-aware GNN Explainer

The explanation adjacency matrix can be expressed by  $A_s(v) = A_c(v) \odot M$ , where  $M$  is a mask matrix, and its entries are binary. Using this expression, directly optimizing (2.10) boils down to an integer problem with respect to  $M$ , and the size of feasible set is  $2^{|\mathcal{E}|}$ . To make the problem tractable, we resort to a relaxation of  $M$  using mean-field variational approximation, and using Monte Carlo method to compute the explanation. As illustrated in Fig. 3, our method sample subgraphs by two steps. First, we decompose the distribution of  $\mathcal{G}$  into a multivariate Bernoulli distribution as  $P(\mathcal{G}) = \prod_{(i,j) \in \mathcal{E}} P(e_{ij})$ , here  $e_{ij} \in [0, 1]$  is the relaxed entry of  $M$ , denoting the probability that edge  $(v_i, v_j)$  is selected. We compute  $e_{ij}$  using an edge embedding network based on the node embeddings. Second, the subgraph is sampled from  $P(\mathcal{G})$  using a reparameterization trick. Since the sampling is not differentiable, we use straight through estimator (STE) to enable the back-propagation. We feed the sampled subgraphs into the GNN with weights frozen, and update the edge embedding network with respect to (2.10).

In detail, we compute the embedding for edge  $(v_i, v_j)$  by a deep neural network. Since our GNN works on undirected graphs, we anonymize the edge direction by defining,

$$\boldsymbol{\mu}_{ij}, \boldsymbol{\Sigma}_{ij} = \frac{1}{2} (\phi(\mathbf{z}_{v_i}, \mathbf{z}_{v_j}) + \phi(\mathbf{z}_{v_j}, \mathbf{z}_{v_i})). \quad (2.11)$$

Similar to variational autoencoder, we learn a mean and a variance for the edge embeddings. In the sampling step, we use the reparameterization trick from the invertible Gaussian family [105],

$$e_{ij} = g(\boldsymbol{\mu}_{ij} + \epsilon * \boldsymbol{\Sigma}_{ij}, \tau), \quad (2.12)$$

here  $\epsilon \sim \mathcal{N}(0, \mathbf{1})$  is a Gaussian noise of the same size  $|\mathcal{E}_c|$ ,  $\tau$  is a temperature parameter,  $g(\cdot, \tau)$  is an invertible smooth function. Specifically, we let,

$$g(y, \tau) = \text{softmax}_{++}(w, \tau) = \frac{\exp(y_k/\tau)}{\sum^{K-1} \exp(y_k/\tau) + \delta}, \quad (2.13)$$

here  $\delta > 0$  ensures the invertibility.  $\lim_{\tau \leftarrow 0} \text{softmax}_{++}(y, \tau)$  is one-hot [105], which allows the straight through gradient estimation. Compared to Gumbel-Softmax, the invertible Gaussian reparameterization is more flexible.

To pursue a compact explanation, we also consider a regularization with respect to the size of the sampled subgraphs. To sum up, we update  $\phi(\cdot)$  with respect to the sampled subgraphs using the following objectives,

$$\min_{\phi} \mathcal{T} = -\mathbb{E}_{Y|\mathcal{G}_S, x_S} [\log P_f(Y|\mathcal{G} = \mathcal{G}_S, X = X_S)] + \lambda \left( \text{ReLU} \left( \sum e_{ij} - K_m \right) \right)^2. \quad (2.14)$$

here  $K_m$  is a predetermined positive integer limiting the node numbers in subgraphs.

### 2.3.4 First-Order v.s. Second-Order

Existing works usually let

$$\mathbf{g} \left( \sum_{u \in \mathcal{N}(v)} \mathbf{m}(\mathbf{h}^{(v)}(t), \mathbf{h}^{(u)}(t)) \right) = \sigma \left( \sum_{u \in \mathcal{N}(v)} \mathbf{w}_u(\mathbf{h}^{(v)}(t) - \mathbf{h}^{(u)}(t)) \right), \quad (2.15)$$

which can be viewed as applying some generalized graph Laplacian to the features. Graph Laplacian matrix associated to a point cloud converges to the Laplace-Beltrami operator on the underlying data manifold [13]. As such, to better capture the data distribution, we should consider both temporal and spatial continuity in GNNs, which corresponds to the depth and the data distributions respectively. We insert the continuous Laplacian operator to (2.3) and abuse  $v$  to represent the hidden state, the first order continuous GNNs can be written as a standard heat equation,

$$\dot{\mathbf{h}}(v, t) = \Delta \mathbf{h}(v, t), \quad \mathbf{h}(v, 0) = \mathbf{h}_0. \quad (2.16)$$

Similarly, the second-order continuous GNNs can be written as an inhomogeneous wave equations,

$$\ddot{\mathbf{h}}(v, t) + \alpha \dot{\mathbf{h}}(v, t) = \Delta \mathbf{h}(v, t) + \mathbf{f}(v, t), \quad (2.17)$$

$$\mathbf{h}(v, 0) = \mathbf{h}_0, \quad \frac{\partial}{\partial t} \mathbf{h}(v, 0) = \mathbf{0}. \quad (2.18)$$

Compared to standard Neural ODEs, the properties of continuous GNNs can be better characterized by the associated partial differential equations. For simplicity, we omit the

weights in GNNS. In the rest of the section, we will use these simplified models to tentatively show that the over-smoothing effect is intrinsic in first-order continuous GNNs, and discuss some pros and cons to use the second-order methods instead of the first-order method.

**Over-Smoothing Effect as Intrinsic Property of First-Order Continuous Graph Neural Networks:** First-order continuous graph neural networks are intimately related to the heat equation, which characterizes the diffusion of heat on a manifold. More specifically, existing works define a Cauchy problem for the homogeneous heat equation. However, the steady-state solution of the heat equation implies that the over-smoothing is intrinsic for first-order continuous graph neural networks. To see this, we have the following proposition:

**Proposition 2.** *Let  $\mathbf{h}_0 \in L^1(\mathbb{R}^n)$ , namely  $\int_{\mathbb{R}^n} |\mathbf{h}_0| dx < \infty$ . For any  $x \in \mathbb{R}^k$ , we have  $\lim_{t \rightarrow +\infty} |\mathbf{h}(x, t)| = 0$ .*

*Proof.* This is a direct result by applying the fundamental solution of heat equation [40]. The fundamental solution is,

$$\mathbf{h}(x, t) = \int_{\mathbb{R}^n} \Phi(x - y, t) \mathbf{h}_0(y) dy, \quad (2.19)$$

$$\Phi(x, t) := \frac{1}{(4\pi t)^{n/2}} \exp\left(-\frac{|x|^2}{4t}\right), \quad (2.20)$$

The heat kernel  $\Phi(x, t)$  is a Gaussian, and we have

$$\begin{aligned} |\mathbf{h}(x, t)| &\leq \frac{1}{(4\pi t)^{n/2}} \int_{\mathbb{R}^n} \exp\left(-\frac{|x - y|^2}{4t}\right) |\mathbf{h}_0(y)| dy \\ &\leq \frac{1}{(4\pi t)^{n/2}} \int_{\mathbb{R}^n} |\mathbf{h}_0(y)| dy. \end{aligned} \quad (2.21)$$

Then we have  $\lim_{t \rightarrow +\infty} |\mathbf{h}(x, t)| = \lim_{t \rightarrow +\infty} \frac{1}{(4\pi t)^{n/2}} \int_{\mathbb{R}^n} |\mathbf{h}_0(y)| dy = 0$ . ■

Using the above result we immediately have  $\lim_{t \rightarrow +\infty} |\mathbf{h}(v_1, t) - \mathbf{h}(v_2, t)| = 0$  for any two nodes  $v_1$  and  $v_2$ . In the context of continuous graph neural networks, it means that node features are eventually blurred when the terminal time goes to infinity. More specifically, the steady-state solution has an exponential decay rate which is not related to the specific form of the original feature distribution  $\mathbf{h}_0$ . This result also implies that the learned data representations are more dispersed to compensate for the exponential decay *w.r.t.* the terminal time if the model is trained using a longer integral time.

**The solution of the second-order CGNN:** The solution of the second-order CGNN is dependent on the initial values. For simplicity, we consider a forced wave equation via setting  $\alpha = 0$  in (2.17) and assume the feature dimension  $n$  is even. First we obtain a homogeneous problem by letting  $\mathbf{f}(v, t) = 0$ . Using the spherical means [40], the solution is,

$$\mathbf{h}(x, t) = \frac{1}{n!!} \left[ \left( \frac{\partial}{\partial t} \right) \left( \frac{1}{t} \frac{\partial}{\partial t} \right)^{\frac{n-2}{2}} \left( \frac{1}{(n+1)\alpha(n+1)} \int_{B(x,t)} \frac{\mathbf{h}_0(y)}{(t^2 - |y-x|^2)^{1/2}} dy \right) \right], \quad (2.22)$$

Here  $B(x, t)$  is a ball,  $\alpha(n+1)$  is the volume of  $n$ -dimension unit ball. Next we insert  $\mathbf{f}(v, t)$  and let  $\mathbf{h}_0(v) = 0$ . We have the following nonhomogeneous problem,

$$\frac{\partial^2}{\partial t^2} \mathbf{h}(v, t) = \Delta \mathbf{h}(v, t) + \mathbf{f}(v, t), \mathbf{h}(v, 0) = 0, \frac{\partial}{\partial t} \mathbf{h}(v, 0) = 0. \quad (2.23)$$

Define  $\mathbf{h}(v, t; s)$  to be the solution of a homogeneous problem with starting time  $s$  and initial value  $\mathbf{h}(v, s) = \mathbf{f}(v, t)$ . Duhamel’s principle asserts that  $\mathbf{h}_n(v, t) := \int_0^t \mathbf{h}(v, t; s) ds$  is the solution to (2.22). The full solution to the forced wave equation is then the sum of (2.23) and  $\mathbf{h}(v, s)$ . The outputs of the second-order model are always related to the initial values since  $\mathbf{h}_0(y)$  is contained in the formulae, which avoids the zero value problem in the first-order case. The solution takes a different form when the feature dimension is odd, but the dependency on initial states are still valid.

**More Discussions:** In this part, we discuss the pros and cons of using second-order continuous GNN instead of the first-order continuous GNN.

Second-order continuous GNN usually learn a smoother transformation than the first-order, which may alleviate over-fitting. The first-order and the second-order continuous GNN both attempt to compute a depth-varying vector field via solving Cauchy problems. However, a severe issue with the first-order Cauchy problem is the stiffness of the learned vector fields. More specifically, the feature mapping  $g$  is a homeomorphism, so the features of Neural ODEs preserve the topology of the input space [36]. Instead, the second-order neural ODEs are not limited to homeomorphic transformations [95].

The first-order augmented ODE implementation of our approach avoids the ad-hoc feature-augmentation in existing first-order graph ODEs and allows better interpretability. To address

this stiffness of learned representations, Augmented Neural ODEs (ANODE) [36] proposes mapping the node features to higher dimensions, which is a widely used trick in Graph-based Neural ODEs. Although augmented neural ODEs can learn high-order dynamics [95], the learned augmented features are difficult to be interpreted, because the high-order behavior is entangled in these features. Our model explicitly defines the augmented features as the gradients of the node features, which have fixed size and clear interpretability.

Sometimes, the second-order models may not be advantageous compared to the augmented first-order models. For example, our definition of the second-order model may not be the minimal augmentation [95]. On the other hand, sometimes, there exist eminent higher-order dynamics behind the data transformation, which can be learned by augmented first-order models. In this case, the second-order models may lose their interpretability and performance superiority.

In general, although it is difficult to claim there exists a performance gap between the optimal potential first-order and the second-order continuous GNN, second-order model has better interpretability, has a simpler form, and usually learns a less stiff data distribution. As such, second-order continuous GNN is more likely to give better performance in practice.

## 2.4 Experimental Results

This section evaluates the performance of our approach on the semi-supervised node and graph classification tasks.

### 2.4.1 Datasets and Experiment Settings

**Semi-supervised Node Classification** For this task, three benchmark datasets are used, including Cora, Citeseer, and Pubmed. We use the standard data splits for the benchmark datasets, where 20 nodes of each class are used for training and another 500 labeled nodes are used for validation. We also include the experiments with random splits. The statistics of datasets are described in Table 1.

**Graph Classification** For this task, we predict the labels for graphs. We consider five datasets [151], including IMDB-Binary (IMDB-B), IMDB-Multi (IMDB-M), COLLAB, Reddit Binary (RDT-B), and Reddit-Multi5k (RDT-M). The statistics of datasets are described in Table 2.

**Explanation for Graph Neural Network** In this task, we follow the setting in GNNExplainer [152] and PGExplainer [81] and construct four kinds of node classification datasets. (1) *BA-Shapes* is a single graph without node features. We first generate a base Barabasi-Albert (BA) graph with 300 nodes. Then we attach 80 “house”-structured graph motifs to the nodes in the base graph randomly. We add 0.1 random edges to perturb the graph. Nodes in the base BA graph is labelled with 0. The top, middle, and bottom nodes of the houses are labelled with 1, 2 and 3 respectively. (2) *BA-Community* is a union of two BA-Shapes graphs with node features. We assign eight classes to the nodes based on the BA-Shapes graph community and the structural roles. We generate node features using two Gaussian distributions for the two BA-Shapes respectively. (3) *Tree-Cycles* is also a base-motif graph. We use an 8-level balance binary tree as the base graph, and attach 80 six-node circle graph motifs to the base graph. (4) *Tree-Grids* uses the same base graph as Tree-Cycles but 3-by-3 grid graph motifs. Table 2 illustrates the synthesized datasets.

**Experimental Settings:** For the classification tasks, we consider three variants of our second-order continuous GNNs: *proposed*, where neither the damping nor the force term is used; *proposed\**, which uses the force term; *proposed\*\**, in which both the damping and the force term are used. In all models we use one linear layer, one continuous GNN layer, and one linear prediction layer. For *proposed*, we use a standard graph convolution layer to approximate the dynamics. For *proposed\**, the force term is approximated by one linear layer with sigmoid function. For *proposed\*\**, the damping term  $\alpha$  is set to 0.95. The terminal time for the continuous layer is set to 15. The hidden dimension is 16. For the prediction layer, we use a dropout rate of 0.2. Cora and Citeseer are trained for 100 epochs using rmsprop optimizer with learning rate 0.005. The rest of the involved datasets are train for 200 epochs using Adam optimizer with learning rate 0.005. All results were collected using single NVIDIA P40 with 24GB GPU memory.

For the explanation task, we follow the quantitative evaluation settings in GNNEx-

plainer [152] and PGExplainer [81]. The explanation is assessed as a binary classification of edges. Specifically, the edges inside motifs are regarded as positive, and negative otherwise.  $e_{ij}$  is viewed the prediction score. We report the average AUC scores and the standard deviations based on ten repeats of the experiments.

## 2.4.2 Baseline Methods

We include representative discrete GNN and continuous GNN methods as the baselines. Given the relation between our second-order method and augmented neural ODEs, we also include related methods. The following results are obtained by run the official implementation of the baselines. When the codes are not available, we cite the values from the original papers. **Discrete GNNs:** For standard discrete GNNs we consider the Graph Convolutional Network (GCN) [64] and the Graph Attention Network (GAT) [130]. Both are the most representative GNN methods, and are considered as widely used baselines for related continuous GNNs methods [103, 146].

**Continuous GNNs:** We include three state-of-the-art continuous GNN models as the related baselines, Graph Neural Ordinary Differential Equations (GDE) [103], Ordinary differential equations on graph networks (GODE) [188], and continuous GNN (CGNN) [146]. We include three variants for GDE with different ODE solvers. For CGNN, we also include two variants, CGNN with weights (WCGNN) and diffusion CGNN.

**Augmented Neural ODEs:** We include Augmented Neural ODEs (ANODE) and Second Order Neural ODEs (SONODE) as baselines. Both ANODE [36] and SONODE [95] are designed for non-structural data, so we only use the node features and drops the graph for these two methods.

## 2.4.3 Comparison Results

### 2.4.3.1 Semi-Supervised Node Classification

: The results for semi-supervised node classification on standard test-train splits are summarized in Table 7 . In most cases, continuous methods outperforms discrete GNNs.



For non-graph methods ANODE and SONODE, the performance is significantly lower than GNNs. The results demonstrates that our approach has superior performance compared to the first-order continuous GNNs in most cases. Particularly, damping term can help improve the performance. Meanwhile, the huge gap between non-graph ANODEs and our approach show that the structural information is critical in node classification, and partial differential equations are of potential. Table 7 summarizes the results for both the predefined and the random test-train splits. Similar observations can be confirmed. It is also observed that GAT is less stable considering the data splits, compared to the continuous methods.

### 2.4.3.2 Graph Classification

: The results for graph classification are summarized in Table 8. Compare to the standard baselines, continuous GNNs are generally superior in performance. Among all continuous methods, our approach with a damping term performs the best. We also notice that the average size of graphs has a large influence on the performance of continuous GNNs. For example, the improvement of continuous methods are much more prominent for RDT-B and RDT-M than for IMDB-B and IMDB-M.

### 2.4.3.3 Prediction Explanation

: The results for prediction explanation are summarized in Table 5. The explanation accuracy is computed based on the ground truth node labels for the synthetic datasets. A better explainability method has higher prediction scores for edges that are in the ground-truth explanation. We compare the explainability of our method to two first-order continuous GNNs. Proposed First-order omit the high order information, which serves as an ablation study. Proposed Second-order is our full method. The quantitative results show that different first order models show similar explainability in most cases. Our second-order method shows some improvement compared to the first order methods.

## 2.4.4 Analysis of Model Parameters

### 2.4.4.1 The model performance v.s. Integration Time

: Figure 4 describes the semi-supervised node classification performance of our method when the model is trained using different integration time. It can be observed that the predicting power is robust to the different integration time. For the first order method, heuristic augmented features are used to achieve a similar phenomenon. Our method suggest a more explicit explanation by defining the high-order behaviors.

### 2.4.4.2 The Effect of Damping Term

: We notice that the damped version of our approach usually yield the best performance, in which  $\alpha$  is a tunable hyperparamater. Figure. 5 describes the semi-supervised node classification performance of our method when the model is trained using different  $\alpha$ . It can be observed that our approach is insensitive to  $\alpha$  for a wide range of  $[0.01, 0.10]$ . When  $\alpha$  is larger (for example, 0.2), the gradients vanishes quickly and the model performance is witnessed to decrease slightly.

## 2.5 Conclusion

In this paper we propose a second order Continuous GNN. Compared to existing methods, our approach employs a second order dynamics, which avoids the over-smoothing and provide additional information to understand the prediction. Our method also can be viewed as a continuous GNN model with interpretable augmented features. Extensive experiments demonstrate that our approach outperforms related baselines for social network applications and has better interpretability.

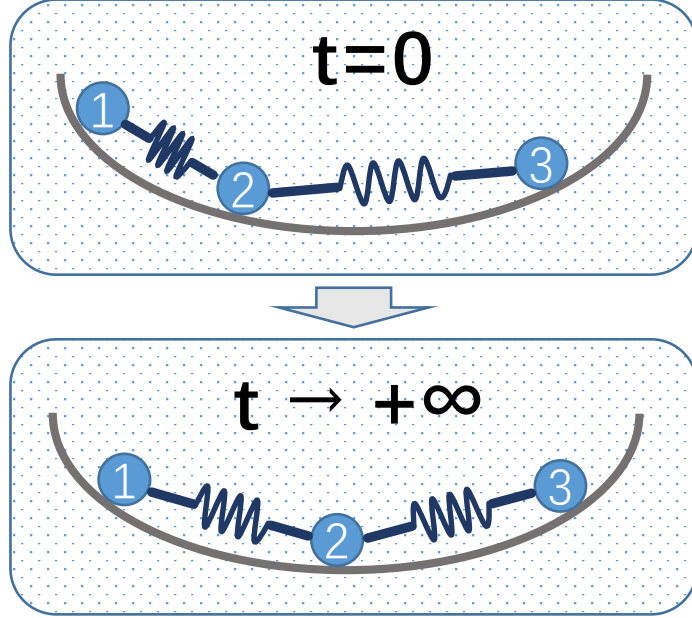


Figure 2: A ball-spring system under the second order dynamics (elastic force, gravity, and friction).

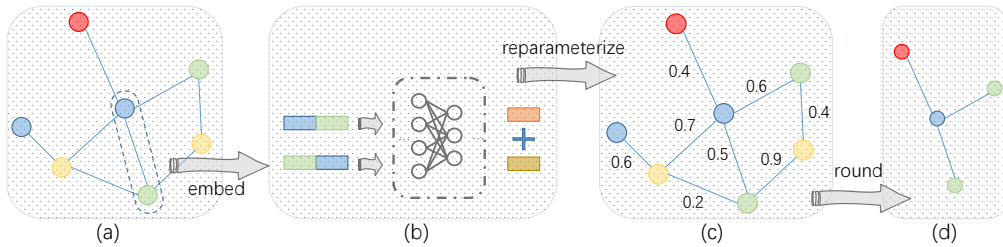


Figure 3: The pipeline for subgraph sampling. We compute the edge embeddings using a deep neural network. We use the invertible Gaussian reparameterization trick to sample the subgraph, which allows the straight-through gradients estimation for the back-propagation. We update the embedding network using Monte Carlo method to optimize the mutual information between the explanation and the computation graph.

Table 1: Statistics of datasets for node classification.

Dataset	# Nodes	# Edges	# Features	# Classes	# Label Rate
Cora	2708	5429	1433	7	0.036
Citeseer	3327	4732	3703	6	0.052
Pubmed	19717	44338	500	3	0.003

Table 2: Statistics of datasets for graph classification.

Dataset	IMDB-B	IMDB-M	COLLAB	RDT-B	RDT-M
# graphs	1000	1500	5000	2000	5000
# classes	2	3	3	2	5
avg.# nodes	19.8	13.0	74.5	429.6	508.5


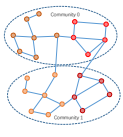



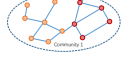


Table 3: Node classification results on citation networks. The Cora, Citeseer, and Pubmed columns summarize the results using standard test-training split. The corresponding columns with superscript \* summarize the results over 10 random splits test-training split. The values for GCN-GODE and GAT-GODE are taken from the original paper. Best-performing methods are bold faced, and the runner-ups are underlined. For the random split, The results of GCN-GODE and GAT-GODE are not available and are denoted by  $-$ .

Dataset	Cora	Citeseer	Pubmed	Cora*	Citeseer*	Pubmed*
GCN [64]	$81.8 \pm 0.8$	$70.8 \pm 0.8$	$80.0 \pm 0.5$	$80.5 \pm 1.1$	$72.3 \pm 0.9$	$79.4 \pm 1.5$
GAT [130]	$82.6 \pm 0.7$	$71.5 \pm 0.8$	$79.7 \pm 0.4$	$80.1 \pm 1.5$	$72.4 \pm 1.2$	$78.4 \pm 1.6$
GDE-rk2 [103]	$82.9 \pm 0.5$	$72.4 \pm 0.5$	$79.8 \pm 0.4$	$81.4 \pm 0.5$	$72.1 \pm 0.6$	$78.9 \pm 0.7$
GDE-rk4 [103]	<u><math>83.8 \pm 0.5</math></u>	$72.4 \pm 0.5$	$79.7 \pm 0.4$	$81.6 \pm 0.4$	$71.8 \pm 0.5$	$79.5 \pm 0.6$
GDE-dpr5 [103]	$81.9 \pm 1.1$	$69.0 \pm 1.1$	$78.3 \pm 0.7$	$79.3 \pm 1.3$	$68.1 \pm 0.9$	$75.1 \pm 0.9$
GCN-GODE [188]	$81.8 \pm 0.3$	$72.4 \pm 0.8$	$80.1 \pm 0.3$	$-$	$-$	$-$
GAT-GODE [188]	$83.3 \pm 0.3$	$72.1 \pm 0.6$	$79.1 \pm 0.5$	$-$	$-$	$-$
CGNN [146]	<u><math>83.8 \pm 1.1</math></u>	$72.7 \pm 0.6$	<b><math>82.2 \pm 0.5</math></b>	$82.5 \pm 1.0$	<u><math>72.2 \pm 1.0</math></u>	<u><math>80.4 \pm 1.3</math></u>
WCGNN [146]	$83.6 \pm 0.7$	$72.8 \pm 0.7$	<u><math>82.1 \pm 0.5</math></u>	$82.0 \pm 1.0$	<u><math>72.2 \pm 0.9</math></u>	$79.8 \pm 0.9$
ANODE [36]	$58.4 \pm 1.5$	$61.6 \pm 0.8$	$69.8 \pm 0.5$	$59.6 \pm 1.3$	$59.9 \pm 1.1$	$70.7 \pm 0.5$
SONODE [95]	$59.9 \pm 1.3$	$61.5 \pm 0.9$	$70.1 \pm 0.4$	$60.1 \pm 1.4$	$60.3 \pm 0.9$	$71.0 \pm 0.6$
Proposed	$83.3 \pm 0.9$	<u><math>72.9 \pm 0.8</math></u>	$82.0 \pm 0.7$	$82.5 \pm 0.8$	<b><math>72.7 \pm 0.6</math></b>	$79.4 \pm 1.0$
Proposed*	$83.5 \pm 0.6$	$72.5 \pm 1.0$	$81.5 \pm 0.5$	<u><math>82.6 \pm 0.6</math></u>	$71.6 \pm 0.8$	$79.9 \pm 0.8$
Proposed**	<b><math>84.3 \pm 0.8</math></b>	<b><math>73.2 \pm 0.9</math></b>	<u><math>82.1 \pm 0.5</math></u>	<b><math>83.5 \pm 1.0</math></b>	$72.1 \pm 0.8$	<b><math>80.9 \pm 0.9</math></b>

Table 4: Graph Classification results.

Dataset	IMDB-B	IMDB-M	COLLAB	RDT-B	RDT-M
GCN [64]	70.9 ± 4.6	<b>49.4 ± 3.2</b>	73.3 ± 3.1	82.9 ± 2.8	52.4 ± 2.1
GDE-rk2 [103]	70.4 ± 3.5	48.5 ± 3.6	75.6 ± 1.6	87.9 ± 1.7	54.1 ± 1.6
GDE-rk4 [103]	71.5 ± 3.8	48.1 ± 2.9	74.9 ± 1.3	86.4 ± 1.5	54.3 ± 1.4
GDE-dpr5 [103]	69.1 ± 4.1	48.2 ± 3.0	71.0 ± 2.5	83.1 ± 3.4	52.0 ± 2.2
CGNN [146]	71.4 ± 3.3	47.4 ± 3.6	75.4 ± 1.9	87.7 ± 2.2	55.3 ± 1.5
WCGNN [146]	71.8 ± 2.9	<u>48.9 ± 2.3</u>	75.7 ± 2.1	<u>88.2 ± 2.3</u>	<u>54.9 ± 1.2</u>
Proposed	71.6 ± 3.1	<b>49.4 ± 3.0</b>	75.2 ± 2.2	88.0 ± 2.0	53.7 ± 1.5
Proposed*	<u>71.9 ± 3.8</u>	48.5 ± 3.4	<u>76.0 ± 1.7</u>	87.5 ± 2.1	53.2 ± 1.4
Proposed**	<b>72.5 ± 3.6</b>	48.8 ± 2.9	<b>76.3 ± 2.5</b>	<b>89.6 ± 1.8</b>	<b>56.0 ± 1.7</b>

Table 5: Explanation AUC results.

	BA-Shapes	BA-Community	Tree-Cycles	Tree-Grid
Base				
Motifs				
GDE-rk2 [103]	<u>0.934</u> ± 0.013	0.877 ± 0.022	0.914 ± 0.010	0.873 ± 0.016
CGNN [146]	0.926 ± 0.017	0.849 ± 0.016	<u>0.955</u> ± 0.015	0.881 ± 0.007
Proposed First-order	0.930 ± 0.015	<u>0.911</u> ± 0.024	0.949 ± 0.008	<u>0.887</u> ± 0.015
Proposed Second-order	<b>0.954</b> ± 0.011	<b>0.955</b> ± 0.021	<b>0.975</b> ± 0.011	<b>0.912</b> ± 0.013

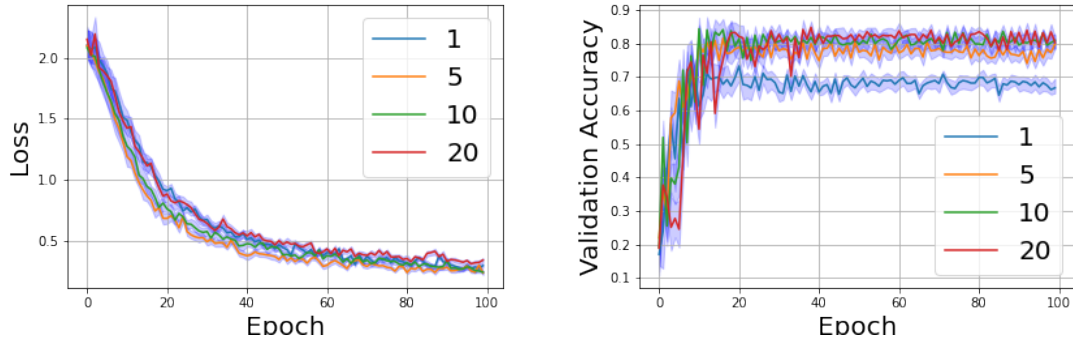


Figure 4: Training loss (left) and validation accuracy (right) under different integration time.

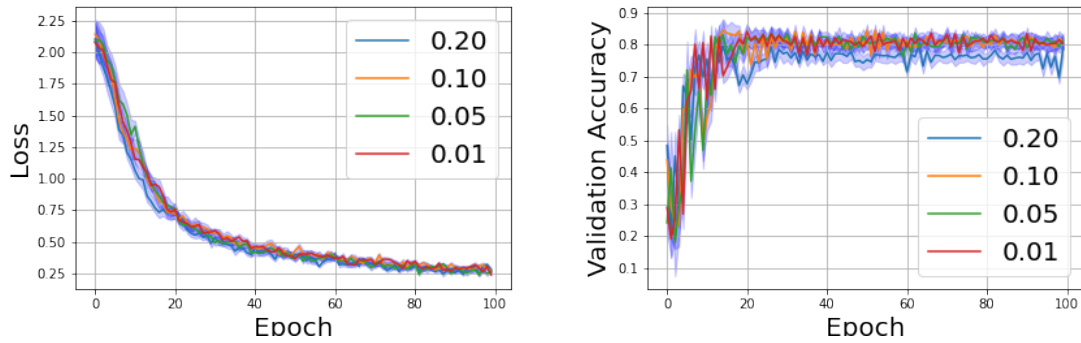


Figure 5: Training loss (left) and validation accuracy (right) with different damping terms.

### 3.0 Robust Self-Supervised Structural Graph Neural Network

#### 3.1 Motivation

Many real-world web based research and applications involve structural data which can be represented using graphs, such as social networks [91], natural languages [79], *etc.* Graph Neural Networks (GNNs) [64, 30, 38, 47, 149, 47, 153, 50, 154] have shown superior performance in learning graph embeddings from structured web data, and the effectiveness has been verified by many downstream tasks, *e.g.* text processing [111, 148], fraud detection [156], recommendation systems [184, 158], and social network link prediction [119, 144]. For example, textGNN [185] extends the twin tower model to employ the user interactions in natural language understanding [128]; in anomaly detection, attention-based GNNs allow users to deduce the root cause of a detected anomaly [31]; in recommendation systems [101], interest aware message-passing can avoid the influence of high-order neighbors with no common interest of a user [77].

By far, most works focus on the analysis for one single graph or a fixed set of graphs. Recently, self-supervised graph representation learning achieved some success both in research and many real web applications [187]. The data sparsity problem is common due to cost of collecting and labelling the data. On the other hand, many problems require domain knowledge, and traditional unsupervised methods lack clear guidance in model designing. Given the success of transfer learning in other machine learning areas, a natural question is the feasibility of generalizable, transferrable, and robust graph representation learning. Inductive representation learning [50] is one of the early works that notice the generalization of GNNs to “unseen” nodes during the learning. Graph Contrastive Coding [107], inspired by the success of contrastive learning, takes one step further to consider the transferability of GNNs. Inspired by self-supervised GNN, there is also evidence showing that the robustness of representations can be enhanced by graph augmentations [157]. At a high level, the prevailing self-supervised GNN pipeline learns the node embeddings via characterizing their local patterns (*i.e.* subgraphs centered at the nodes of interest) and makes use of the instance



discrimination [145] framework to learn the relationships between these embeddings. There are some success attempts to apply self-supervised learning to real problems. For example, in recommendation systems, the quality of users and items representations from historical interactions is central to the success of collaborative filtering. Via incorporating item contents into the learning scheme, the contrastive-based methods can enforces dimension-wise similarity between feature representation and collaborative embedding, which avoid some noise edges, such as the irrelevant interaction of users with a large bulk of items. With the dramatic social media boom, the social recommender system can find the high-order connectivity information from the social influential users using self-supervised methods [158]. and capture the behavior of users even for those having few interactions with items. In fraud detection in finance and spammer discovery in social media, the hop-count distribution is different for anonymous nodes and normal nodes, and predicting shortest path length between a pair of nodes using self-supervised learning methods can provide some evidence to anomaly detection [156].

Although the success of aforementioned self-supervised methods in social networks and related applications, several important problems remain to address. Firstly, the node representation learned in most existing self-supervised methods is focused on the node-wise proximity, and the proximity of local structures is less considered. Secondly, the local consistency of the users are seldom considered, which is related to distribution shift. Sub-graph sampling techniques are widely exploited in the training of GNNs as a consequence of the size of modern large-scale graph data. However, additional distribution shifts would be introduced by different sub-graph sampling techniques, while prior works implicitly assume that the data distribution is universal for different sources. The distributional shift potentially deteriorates the generalization and transferability. These problems urge *network proximity defined on the embedding of local structures* and invoke an explicit consideration on the *distributional robustness* of the self-supervised GNNs.

To address the aforementioned challenging problems, in this paper, we proposed a novel distributionally robust graph contrastive learning framework, dubbed learning *structural embeddings*. In contrast to existing works that focus on the node-wise proximity, our method learns the local-structural level proximity via gathering an embedding set describing both a node and its neighbors, *i.e.* a subgraph surrounding the node of interest. The

difference between two nodes is then characterized via the Wasserstein distance defined on the distribution of nodes of corresponding subgraphs. To alleviate the distributional shift, we further design a distributionally robust contrastive learning framework by constraining the maximal inconsistency for similar subgraphs. The Wasserstein distance to measure the proximity for self-supervised learning is considerably challenging in computation because the contrastive loss function based on the Wasserstein distance becomes a difficult bi-level optimization problem. In this paper, we employ a differentiable implicit layer to deal with the Wasserstein distance, making our framework end-to-end trainable. Our contributions are summarized as follows:

- We propose new graph embeddings at a local-structural level via learning the embedding sets for the subgraphs, which are composed of the nodes of interest and their neighbors. The Wasserstein distance is adopted as the network proximity.
- We introduce a distributionally robust contrastive learning framework. To circumvent the difficult minimax problem, the original problem is relaxed to an asymptotic formulation. We resort to the differentiable optimization methods to compute the Wasserstein distance, which makes the full network end-to-end trainable.
- Extensive experiments are conducted on various tasks and representative datasets. The results demonstrate that our algorithm outperforms other state-of-the-art methods in several important downstream analysis. The ablation study validates the effectiveness of our approach.

### 3.1.1 Notations

Throughout the paper, the bold capital and bold lowercase symbols are used to represent matrices and vectors, respectively. If all elements of a matrix  $\mathbf{A}$  are greater than or equal to 0, we denote it by  $\mathbf{A} \geq 0$ . We use  $G = \{V, E\}$  to represent a graph. Here  $V$  is the node set, and  $E$  is the edge set. Finally, a  $n \times n$ -identity matrix is denoted by  $\mathbf{I}_n$ ,  $\mathbf{1}_n$  is a  $n$ -dimension one vector, and  $\mathbf{0}$  denotes a zero matrix.

## 3.2 Related Works

### 3.2.1 Graph Representation Learning

Graph representation learning is featured by mining the topological structures of graphs and encoding nodes with low-dimensional embeddings. Representative works, including Word2Vec [87], DeepWalk [102], and LINE [124], collect local patterns and learn mappings from graphs to vectors. To capture the highly non-linear property of attributed graphs, Deep Attribute Embedding Network (DANE) [45] upgrades the shallow models in the aforementioned model to deep networks.

There are increasing interest in GNN [64], which is developed from graph convolution and can simultaneously exploit the structural knowledge and the enriched side information in attribute graphs. GCN [64] shows the superiority of the first-order graph convolution for semi-supervised node classification. GraphSage [153, 50, 154] and Message Passing Neural Network (MPNN) [47] broaden the application of GNNs to the analysis for large-scale graphs via a more efficient aggregation mechanism. Graph Attention Network (GAN) [130] introduces the attention mechanism which is shown to be effective in general network analysis. recently, the relation between GNNs and graph isomorphism problem is studied, based on which Graph Isomorphism Network (GIN) [149] is designed.

In this paper we use GIN as the backbone models in our self-supervised graph neural networks.

### 3.2.2 Graph Contrastive Learning

Recently, contrastive learning has attracted a surge of attention in a machine learning community. A typical scheme is to select an anchor, a positive instance, and a negative instance, and maximize the margin between the similar pair (anchor *v.s.* positive) and the negative pair (anchor *v.s.* negative). Instance discrimination [145] is a popular self-supervised framework that achieves state-of-the-art performance in many computer vision tasks, for example, SimCLR [26] and SimCLRv2 [27].

Graph contrastive learning requires different objectives due to the unique data structure,

such as Jensen-Shannon estimator [132, 118], the noise-contrastive estimation (NCE) [157] and parametric estimation method using projection head [52]. To generate graph views, different augmentation methods are proposed. One of the most common method is node attribute masking [157] which applies the feature transformations. For a given graph, edge perturbation [157, 107] can randomly adds or drops edges. Another direction is sampling-based transformation. For example, ego-nets sampling, such as in DGI [132], InfoGraph [118] and MVGRL [52], can be viewed as the unification of the contrast between graph-level and node-level representations.

In this work, we adopt the InfoNCE loss [97] and instance discrimination [145] methods, which are demonstrated to have good performance in graph pre-training [107].

### 3.2.3 Network Proximity

There are several perspectives concerning the definition for the proximity between different nodes in networks. A variety of works are based on the neighborhood similarity computed locally from the neighborhood of the vertices, for example, Jaccard similarity and cosine similarity. On the other hand, the structural similarity consider the similarity *w.r.t.* local patterns. Models of this genre include structural diversity [127], motif [88, 15], and spectral methods [34]. A more fine-grained definition [124] considers the first-order proximity, second-order proximity, and high-order proximity. GraRep [21] considers the connectivity between different nodes via explicitly constructing the probability transition matrix. Alternatively, DeepWalk [102] explores the connectivity and the local pattern via random walk with restart. For attribute graphs, GNN [64] and its variants [153, 130] are powerful tools to leverage the side information in computing network proximity. Another related topic to the network proximity is the graph matching problem, where a principally similar idea to the graph neural network is developed. The representative works in this direction include the Weisfeiler-Lehman Isomorphism Test [113] and the related graphlet methods [114].

### 3.2.4 Deep Implicit Layers

Deep neural networks are heavily dependent on the gradient-based optimization, *e.g.* Momentum Stochastic Gradient Descent [120] and Adam [63]. However, constrained optimization is seldom integrated into deep neural networks due to the difficulty of the automatic differentiation concerning the boundary.

Recently, neural ordinary differential equations [25] provides a new interpretation for the residual neural layers, which states that each residual layer can be viewed as one differential operation. Inspired by this perspective, it is shown that deep neural networks can be utilized to solve convex constrained problems, referred to as differentiable optimization [4]. OptNet [6] is an initial study, in which a special deep layer is designed to solve quadratic programming problems. Of note, in regular tasks, the parameters in the convex constrained problems of interest usually are computed using the outputs from preceding layers. An important result of the deep implicit layer is that the gradient of the parameters can be computed using the implicit function theorem.

In this paper, we use Wasserstein distance to characterize the node proximity, which can be formulated as a linear programming problem. Given the success of deep earth move distance in few-shot learning [162], we adopt the deep implicit layer to compute Wasserstein distance, which admits a fully end-to-end graph neural network.

## 3.3 Proposed Method

At a high level, we sample subgraphs spanned from a node and learn the embeddings via a graph encoder to capture the local pattern. Each node is an individual class and the similarity of subgraphs is characterized by the network proximity defined on the embeddings. Figure 6 illustrates the pipeline of our approach. We consider a subgraph triplet  $(G_{q_i}, (G_{k_i}, G_{k_j}))$ , in which  $G_{q_i}$  and  $G_{k_i}$  are from the same node and  $G_{k_j}$  is from a different one. We encourage a large margin between the proximity for similar pair  $(G_{q_i}, G_{k_i})$  and that for dissimilar pair  $(G_{q_i}, G_{k_j})$ .

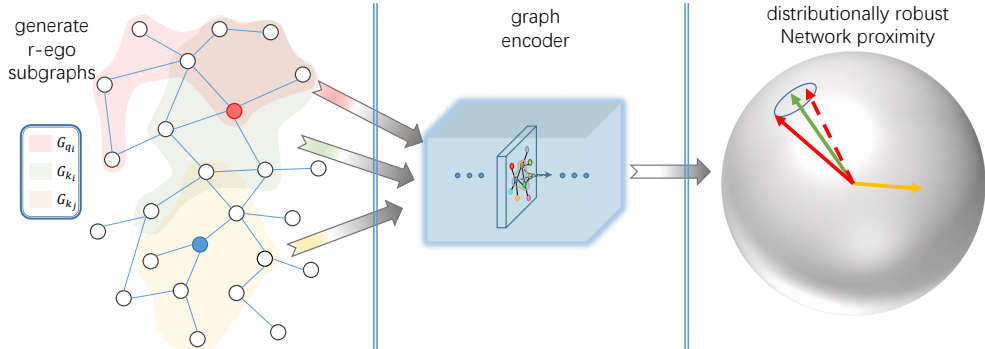


Figure 6: Subgraphs generated from the same node (colored in red) are similar, and from different nodes are dissimilar (red *v.s.* blue). To learn a robust encoder, we fix the keys, *i.e.*  $G_{k_i}$  and  $G_{k_j}$ , and focus on the most difficult query  $G_{q_i}$  (solid red line) instead of random queries (dashed red lines).

Compared to prior works, our approach is robust in two senses. Firstly, we expand the embedding space and exploit Wasserstein distance as the network proximity. Sampling subgraphs is a standard step for modern large-scale graph analysis. In this paper, we propose to use *structural embeddings* defined on subgraphs crawling around the nodes of interest, *i.e.* the comprehensive information gathered from the center nodes and their neighbors. We use the structural embeddings, which are sets of node embeddings, to explicitly capture the subtle difference between subgraphs. More specifically, we characterize the non-linear and non-local relationships between these embeddings by Wasserstein distance. Of note, we propose to utilize an automatic differentiable solver to compute the Wasserstein distance, which leads to an end-to-end trainable network. Secondly, we focus on the most ambiguously similar pairs, which leads to our formulation insensitive to data distributions. For example, sampling techniques and their parameters may introduce fluctuation concerning the distribution of subgraphs. We propose an asymptotically distributionally robust contrastive learning framework, which is reluctant to distribution shift and easy for computation.

In the following, we first formalize our self-supervised GNN and the associated network proximity in §3.3.1. Then, in §3.3.2 we propose our distributionally robust contrastive learning

framework and formulate an asymptotic scenario to make the problem tractable. At last, in §3.3.3 we introduce an end-to-end trainable neural network featured by a differentiable implicit layer to compute the Wasserstein distance. We summarize the full algorithm and discuss some details in §3.3.4.

**Notations:** we use the bold capital and bold lowercase symbols to represent matrices and vectors.  $\mathbf{A} \geq 0$  denotes all elements of a matrix  $\mathbf{A}$  are greater than or equal to 0.  $G = \{V, E\}$  is a graph,  $V$  is the node set, and  $E$  is the edge set. A  $n \times n$ -identity matrix is denoted by  $\mathbf{I}_n$ ,  $\mathbf{1}_n$  is a  $n$ -dimension one vector, and  $\mathbf{0}$  denotes a zero matrix.

### 3.3.1 Self-Supervised Graph Neural Network via Wasserstein Proximity

We adopt the  $r$ -ego network [107] to represent the local structure of node  $i$ , defined as follow,

**Definition 1.  $r$ -ego network [107]** For graph  $G = (V, E)$ , the  $r$ -neighbors of a node  $v \in V$  are defined as  $\mathcal{N}_v = \{u | d(u, v) \leq r\}$ , where  $d(u, v)$  is the shortest path length between node  $u$  and  $v$ . The  $r$ -ego network of  $v$ , denoted by  $G_v$ , is the subgraph induced by  $\mathcal{N}_v$ .

$r$ -ego networks can be augmented via graph sampling techniques, *e.g.* random walks with restart [125]. We consider two subgraphs via augmenting the same node  $i$ , denoted as  $G_{q_i}$  and  $G_{k_i}$ , and other subgraphs from different nodes  $\{j\} \subset V$ , denoted as  $\{G_{k_j}\}$ . In instance discrimination learning, each  $r$ -ego network is viewed as a distinct instance. Therefore,  $G_{q_i}$  and all  $G_{k_j}$  are considered to be similar, and  $G_{q_i}$  and  $G_{k_i}$  are considered to be dissimilar. Specifically,  $G_{q_i}$  is referred to as the *query*, and the elements in the set  $\{G_{k_j}\} \cup \{G_{k_i}\}$  are referred to as the *keys*.

Our self-supervised GNN are then encouraged to maximize the margin between similar instances and dissimilar instances. We define  $f_q, f_k : V \times \mathcal{S} \rightarrow \mathcal{Y}$ , where  $\mathcal{S}$  is the augmentation function space and  $\mathcal{Y}$  is the embedding space.  $f_q(\cdot)$  and  $f_k(\cdot)$  map an augmentation of a node to the query embeddings and the key embeddings respectively, and for simplicity we omit the augmentation function. Let  $d_g : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$  be a network proximity function, then  $d_g(f_q(i), f_k(j))$  represents the proximity between query node  $i$  and key node  $j$ . Finally, let  $\ell_d(\cdot)$  be the objective function for our self-supervised graph neural network, which is defined

on the proximity between a group of nodes and detailed in the following.

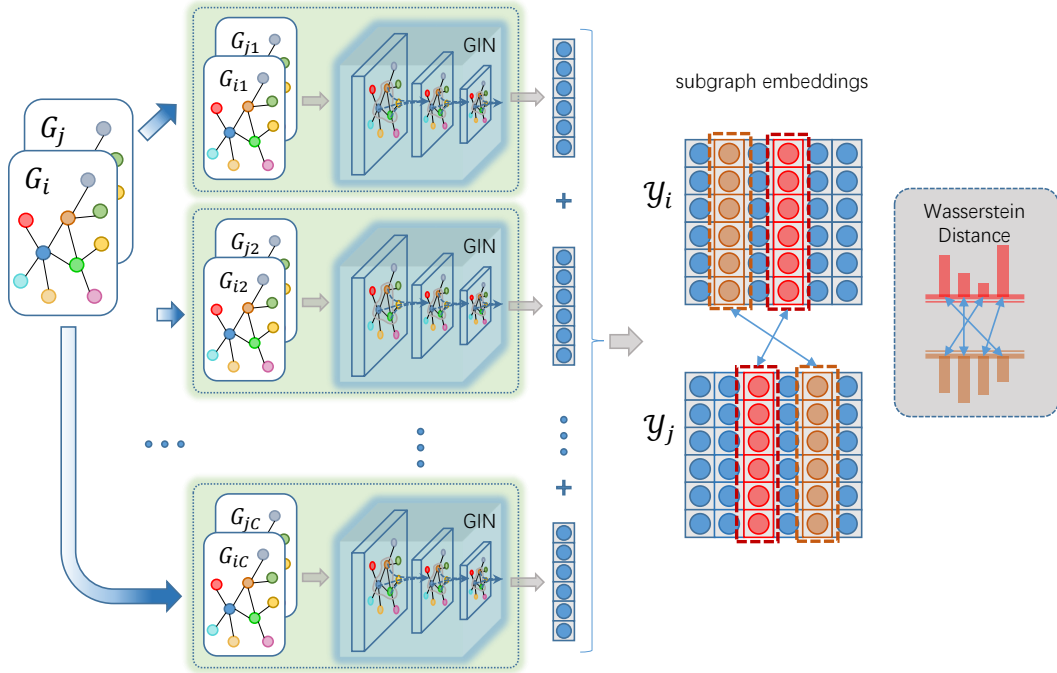


Figure 7: The  $r$ -ego subgraphs surrounding the nodes of interest are fed into the structural GNN to obtain the embedding sets. Wasserstein distance is then computed as the network proximity.

Prior works usually defines  $\mathcal{Y} \subseteq \mathbb{R}^n$ , *i.e.* a  $n$ -dimension vector space. In this paper, we propose to enlarge the capacity of the embedding space by defines  $\mathcal{Y} \subseteq \mathbb{R}^{n \times m}$ . The computation of our method is illustrated in Fig. 7.

In the sampling framework, the node representations are computed on the induced subgraph instead of the full graphs, in which potential bias is introduced due to the perturbation in the sampled  $r$ -ego subgraph  $G_i$ . To address this problem, an alternative method is to expand the node representations to *subgraph representations* which allows some uncertainty. More specifically, we use an *embedding set* for  $r$ -ego subgraph  $G_i = (V_i, E_i)$ , instead of the fixed vector representation. Let  $\mathbf{y}_{ic}$  be the embedding for a node  $v_{ic} \in G_i$ , the embedding set for  $G_i$  can be denoted as  $\mathcal{Y}_i = \{\mathbf{y}_{ic} | c = 1, 2, \dots, C\}$  with  $C$  be the cardinality of  $V_i$ . To learn an embedding set, we can use a two-stage subgraphs sampling on top of a general graph



neural network backbone: given the node  $i$ , in the first stage, we sample  $G_i$  to obtain  $V_i$ , and in the second stage we sample  $\bar{G}_i = \{G_{ic} | c = 1, 2, \dots, C\}$ . All  $G_{ic}$  are fed into the backbone model and the node embeddings are gathered to form a subgraph embedding.

The embedding sets are composed of multiple node representations therefore having better representation abilities and larger capacity for capturing the subtle difference in sampled subgraphs. The network proximity defined on the embedding sets can be interpreted as a matching problem. We adopt the 1-Wasserstein distance to evaluate the difference between  $\mathcal{Y}$  and  $\tilde{\mathcal{Y}}$ ,

$$\mathcal{W}(\mathcal{Y}, \tilde{\mathcal{Y}}) = \min_{x_{ij}} \sum_{i,j=1}^C x_{ij} c(\mathbf{y}_i, \tilde{\mathbf{y}}_j), \quad c(\mathbf{y}_i, \tilde{\mathbf{y}}_j) = 1 - \frac{\mathbf{y}_i^\top \tilde{\mathbf{y}}_j}{\|\mathbf{y}_i\| \|\tilde{\mathbf{y}}_j\|}, \quad (3.1)$$

here  $x_{ij}$  is a match and we reserve the constraints concerning  $x_{ij}$  for the next section. The cost function  $c(\mathbf{y}_i, \tilde{\mathbf{y}}_j)$  measures the dissimilarity between  $\mathbf{y}_i$  and  $\tilde{\mathbf{y}}_j$ .

### 3.3.2 (Asymptotically) Distributionally Robust Contrastive Learning

#### 3.3.2.1 Distributionally Robust Contrastive Learning

: The support of the embeddings  $\mathcal{Y}_i$  for node  $i$  is exactly the feasible set for  $r$ -ego subgraphs  $G_i$ . Prior works consider an empirical expectation form for the objective functions. However, as aforementioned, the distributions of  $r$ -ego subgraphs are presumably affected by the specific sampling method. Local structures potentially lead to significantly different distributions for  $r$ -ego subgraphs, and an example is given in Fig. 8. As such, it may introduce bias into the learned global information. To avoid this bias, robustness is desired for our self-supervised graph neural networks.

In this paper, we focus on the difficult queries instead of equally treating all queries. Specifically, for given keys and candidate  $r$ -ego subgraphs, we only consider the most difficult query, defined as the one that has the worst similarity with the matched key. Formally, we define,

$$\mathcal{L} = \min_{f_q, f_k} \max_{G_{q_i}} -\frac{1}{N} \sum_{i=1}^N \log \frac{1}{Z} \exp(\mathcal{W}_{\pi_i}(G_{q_i}, G_{k_i})), \quad (3.2)$$

here  $\mathcal{W}_{\pi_i}$  is Wasserstein distance dependent on a data-related constrain  $\pi_i$  whose details will be discussed in the next section.  $\mathcal{Z}$  is a normalization term,

$$\mathcal{Z} = \exp(\mathcal{W}_{\pi_i}(G_{q_i}, G_{k_i})) + \sum_{j=1}^m \exp(\mathcal{W}_{\pi_j}(G_{q_i}, G_{k_j})). \quad (3.3)$$

We refer (3.2) as the *distributionally robust contrastive learning*, because a contrastive objective is adopted and the it can be represented as a distributionally robust optimization (DRO) problem.

The distributionally robust formulation is weakly related to the support of the embedding set  $\mathcal{Y}_i$ . Rather, compared to the expectation form, it directly constrains the most diverse match between the query and key. This property assures the learned model is *robust* to distribution shift caused by either the sampling methods for  $r$ -ego subgraphs or the data difference.

### 3.3.2.2 Asymptotically Distributionally Robust Contrastive Learning

: The DRO problem defined by (3.2) can be solved via the duality form. In detail, a feasible method is to solve the inner maximal problem and the outer minimal problem alternatively, for example, the projected gradient descent in adversarial training. However, in our case the inner problem is difficult and the reasons are two-folded. Firstly, the closed-form solution is difficult to derive, due to the complexity of the subgraph distributions and the computation of the Wasserstein distance. Secondly, the graph data are structural and discrete, which means that gradient-based methods cannot be immediately adopted. To overcome the challenge, we propose an asymptotic relaxation.

For simplicity, let  $\ell_w(G_{q_i}) = \log \frac{1}{\mathcal{Z}} \exp(\mathcal{W}_{\pi_i}(G_{q_i}, G_{k_i}))$ . We re-write (3.2) as follow,

$$\max_{G_{q_i}} \ell_w(G_{q_i}) = \sum (\mathbb{I}_i(G_{q_i}) \ell_w(G_{q_i})) \quad (3.4)$$

here  $\mathbb{I}_i(G_{q_i})$  is an indicator function,

$$\mathbb{I}_i(G_{q_i}) = \begin{cases} 1, & G_{q_i} = \arg \max_{G_{q_i}} \ell_w(G_{q_i}) \\ 0, & \textit{otherwise} \end{cases}.$$

Problem (3.2) is transformed into an integer optimization problem. To make the problem tractable, we relax  $\mathbb{I}_i(G_{q_i})$  with  $h_i(G_{q_i})$ ,

$$h_i(G_{q_i}) = \left( 1 + \exp \left( \frac{1}{\tau} \left( \hat{\mathcal{W}}_i - \mathcal{W}_{\pi_i}(G_{q_i}, G_{k_i}) \right) \right) \right)^{-1}, \quad (3.5)$$

here  $\tau$  is a temperature parameter,  $\hat{\mathcal{W}}_i$  is threshold indicating the maximal Wasserstein distance. The relaxation is referred to as *asymptotically Distributional robustness* because when  $\tau \rightarrow 0$  and  $\hat{\mathcal{W}}_i \rightarrow \max_{G_{q_i}} \mathcal{W}_{\pi_i}(G_{q_i}, G_{k_i})$ ,  $h_i(G_{q_i})$  converges to the distributionally robust objective.  $\hat{\mathcal{W}}_i$  can be empirically estimated and updated during the optimization, and the details are given in §3.3.4.

### 3.3.3 Computing Wasserstein Distance via Deep Implicit Layer

(3.2) involves the computation of the 1-Wasserstein distance between the queries and the keys, which can be formulated as a linear programming problem,

$$\begin{aligned} \mathcal{W}_{\pi_i}(G_1, G_2) &= \min_{x_{ij}} \sum_i \sum_j c_{ij} x_{ij}, & (3.6) \\ \text{s.t. } x_{ij} &\geq 0, \quad \sum_i x_{ij} = s_j, \quad \sum_j x_{ij} = d_i, \quad \forall i, j, \end{aligned}$$

here  $c_{ij}$ ,  $s_j$  and  $d_i$  are parameters computed from the embeddings, which can be viewed as functions defined on some input  $\theta$ . In our case,  $\theta$  can be interpreted as the embeddings and other model parameters. Directly integrating (3.6) in our deep model will lead to intractable gradients. To solve this challenged problem, we resort to a deep implicit layer encoding the Wasserstein distance computation. A deep implicit layer exploits the (Karush–Kuhn–Tucker) KKT conditions to solve a convex problem with the parameters fixed, and computes the gradients *w.r.t.*  $\theta$  using the implicit function theorem. As such, we can build a deep neural network by integrating the graph encoder and the Wasserstein layer, which can solve (3.6) and can be trained in an end-to-end manner. For self-containedness, the detailed derivation of the deep implicit layer encoding the Wasserstein distance is given in the following.

Let  $N = m \times n$ ,  $f(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{c}^\top \mathbf{x}$ ,  $\mathbf{x} \in \mathbb{R}^N$ , and the  $k$ -th entry is  $x_{ij}$  with  $i = \lfloor k/n \rfloor$  and  $j = k \bmod n$ .  $\mathbf{c} : \Theta \rightarrow \mathbb{R}^N$  is the vectorized form of the cost function. We first re-write (3.6) as,

$$\begin{aligned} \mathcal{W}_{\pi_i}(G_1, G_2) &= \min f(\mathbf{x}, \boldsymbol{\theta}), \\ \text{s.t.} \quad G(\boldsymbol{\theta})\mathbf{x} &\leq \mathbf{0}, \quad h(\mathbf{x}, \boldsymbol{\theta}) = 0, \end{aligned} \tag{3.7}$$

here  $G(\boldsymbol{\theta}) = \text{diag}(-\mathbf{1})$ , and  $\text{diag}(\cdot)$  maps the given vector to a diagonal matrix. Let  $\mathbf{s} \in \mathbb{R}^m$  and  $\mathbf{d} \in \mathbb{R}^n$  be the vectorized  $s_j(\boldsymbol{\theta})$  and  $d_i(\boldsymbol{\theta})$ .  $h(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{A}\mathbf{x} - [\mathbf{s}, \mathbf{d}]$ , where  $[\mathbf{s}, \mathbf{d}]$  is the concatenation of  $\mathbf{s}$  and  $\mathbf{d}$ .  $\mathbf{A} \in \mathbb{R}^{N \times (n+m)}$  arranges the linear constraints, , defined as,

$$\mathbf{A} = \begin{bmatrix} \mathbf{1} & & & \\ & \ddots & & \\ & & \mathbf{1} & \\ \text{diag}(\mathbf{1}) & \cdots & \text{diag}(\mathbf{1}) & \end{bmatrix}. \tag{3.8}$$

(3.7) is a constrained convex problem. Let  $\mathbf{v}$  and  $\boldsymbol{\lambda}$  be the dual variables for the equalities and inequalities in the constraints, its Lagrangian is,

$$L(\mathbf{x}, \mathbf{v}, \boldsymbol{\lambda}, \boldsymbol{\theta}) = \mathbf{c}^\top \mathbf{x} + \boldsymbol{\lambda} G(\boldsymbol{\theta})\mathbf{x} + \mathbf{v} h(\mathbf{x}, \boldsymbol{\theta}). \tag{3.9}$$

It is easy to verify that (3.7) satisfies the Slater's condition and the twice differentiability. As such, the KKT conditions of (3.9) are the necessary and sufficient optimality conditions for (3.7). More specifically, define  $g(\mathbf{x}, \mathbf{v}, \boldsymbol{\lambda}, \boldsymbol{\theta}) = [\nabla_{\mathbf{x}} L(\mathbf{x}, \mathbf{v}, \boldsymbol{\lambda}, \boldsymbol{\theta}), \text{diag}(\boldsymbol{\lambda})G(\boldsymbol{\theta})\mathbf{x}, h(\mathbf{x}, \boldsymbol{\theta})]^\top$ . If  $g(\tilde{\mathbf{z}}, \boldsymbol{\theta}) = 0$  for some  $\tilde{\mathbf{z}} = (\tilde{\mathbf{x}}, \tilde{\mathbf{v}}, \tilde{\boldsymbol{\lambda}})$  where  $\tilde{\mathbf{x}}$  and  $\tilde{\mathbf{v}}$  are both feasible, then the KKT conditions are satisfied and  $\tilde{\mathbf{x}}$  is optimal. We can compute the partial Jacobian regarding  $\mathbf{z}$  (omitting the last two columns) and  $\boldsymbol{\theta}$  as,

$$D_{\mathbf{z}} g(\tilde{\mathbf{z}}, \boldsymbol{\theta}) = \begin{bmatrix} D_{\mathbf{x}} \nabla_{\mathbf{x}} L(\tilde{\mathbf{z}}, \boldsymbol{\theta}) \\ \text{diag}(\tilde{\boldsymbol{\lambda}}) D_{\mathbf{x}} G(\boldsymbol{\theta}) \tilde{\mathbf{x}} \\ D_{\mathbf{x}} h(\mathbf{x}, \boldsymbol{\theta}) \end{bmatrix}, D_{\boldsymbol{\theta}} g(\tilde{\mathbf{z}}, \boldsymbol{\theta}) = \begin{bmatrix} D_{\boldsymbol{\theta}} \nabla_{\mathbf{x}} L(\tilde{\mathbf{z}}, \boldsymbol{\theta}) \\ \text{diag}(\tilde{\boldsymbol{\lambda}}) D_{\boldsymbol{\theta}} G(\boldsymbol{\theta}) \tilde{\mathbf{x}} \\ D_{\boldsymbol{\theta}} h(\mathbf{x}, \boldsymbol{\theta}) \end{bmatrix}. \tag{3.10}$$

The following theorem characterizes the differentiability of convex optimization.

**Theorem 1.** (*Differentiability of a Convex Optimization Problem [9]*) Given a convex problem, assume (1) Slater condition holds, (2) all derivative exists, (3)  $\{i|\lambda_i = 0 \text{ and } f_i(x,\theta)\} = \emptyset$ , and (4)  $D_{\mathbf{x}}g(\mathbf{x}, \mathbf{v}, \boldsymbol{\lambda}, \boldsymbol{\theta})$  is non-singular. If  $g(\tilde{\mathbf{x}}, \tilde{\mathbf{v}}, \tilde{\boldsymbol{\lambda}}, \boldsymbol{\theta}) = 0$ , the solution mapping has a single-valued localization  $s$  around  $\tilde{\mathbf{x}}, \tilde{\mathbf{v}}, \tilde{\boldsymbol{\lambda}}$  that is continuously differentiable in a neighborhood  $\mathcal{Q}$  of  $\boldsymbol{\theta}$  with Jacobian satisfying,  $D_{\boldsymbol{\theta}}s(\boldsymbol{\theta}) = -D_{\mathbf{z}}g(\tilde{\mathbf{x}}, \tilde{\mathbf{v}}, \tilde{\boldsymbol{\lambda}}, \boldsymbol{\theta})^{-1}D_{\boldsymbol{\theta}}g(\tilde{\mathbf{x}}, \tilde{\mathbf{v}}, \tilde{\boldsymbol{\lambda}}, \boldsymbol{\theta})$  for every  $\boldsymbol{\theta} \in \mathcal{Q}$ .

**Remark 1.** Theorem 1 is an immediate result from the Implicit Function Theorem [66]. Recall that in our problem,  $\boldsymbol{\theta}$  are the embeddings of subgraphs. Theorem 1 states that the gradient w.r.t.  $\boldsymbol{\theta}$  can be computed according to (3.10). In other words, backward propagation is feasible.

### 3.3.4 Algorithm and Implementation

---

**Algorithm 1** Distributionally Robust Self-Supervised Graph Neural Network

---

**Input:** Graph  $G$ , key size  $C$

**Output:** Model Parameter  $W$

```

1 Initialize  $\tau, \hat{\mathcal{W}}_i$ ;
2 while epoch not end do
3   for  $i \in V$  do
4     Sample  $r$ -ego subgraphs,  $G_{q_i}$  and  $G_{k_i}$  ;
5     Sample  $r$ -ego subgraphs,  $G_{k_j}, j = 1, 2, \dots, C$ ;
6     Update  $W$  w.r.t. (3.4) and (3.5), ;
7      $\hat{\mathcal{W}}_i \leftarrow 0.999 \times \max \left\{ \hat{\mathcal{W}}_i, \mathcal{W}_{\pi_i}(G_{q_i}, G_{k_i}) \right\}$  ;
8      $\tau \leftarrow 0.999 \times \tau$ ;
9   end
10 end
```

---

The first stage of sampling is based on neighbor sampling, and the second stage is based on random walk with restart. To accelerate the training, we use the one-hop neighbors in the first stage. We use Graph Isomorphism Network (GIN) as the backbone model in our structural GNN. For the Wasserstein distance, let  $\mathbf{u}$  and  $\mathbf{v}$  be two embedding sets,  $s_j$  in (3.6)

is defined by,

$$s_j = \frac{C\bar{s}_j}{\sum_{i=1}^C \bar{s}_j}, \text{ where } \bar{s}_j = \max \left\{ \sum_{k=1}^C \mathbf{u}_j^\top \mathbf{v}_k, 0 \right\}, \quad (3.11)$$

and  $d_i$  is defined similarly. We initialize  $\hat{\mathcal{W}}_i$  and  $\tau$  in (3.5) with 0 and 3 respectively.  $\hat{\mathcal{W}}_i$  is estimated dynamically and  $\tau$  is gradually decreasing during training. The full algorithm is summarized in Algo. 1.

Our implementation is based on the DGL package <sup>1</sup> and OptNet [6], which is designed for solving quadratic programming (QP) problems. Specifically, the backward propagation is accomplished via automatic differentiation through a customized QP solver <sup>2</sup>. To tailor our problem to fit into OptNet, we only need to substitute the objective function  $\mathbf{c}^\top \mathbf{x}$  with a quadratic form  $\frac{1}{2} (\mathbf{c}^\top \mathbf{x})^2$ , and the above derivation remains untouched. It is easy to verify that the surrogate quadratic objective has the same optimums as the original linear programming problem (3.6).

### 3.4 Experimental Results

In this section, we evaluate our approach for downstream graph analysis, which includes two steps. We first pre-train the model using several large-scale graph datasets. Then we finetune the pre-trained model according to the specific tasks. In §3.4.1 we describe the pre-training setting of our robust self-supervised graph neural network. In §3.4.2 we detail the finetuning and evaluate our approach on three standard downstream benchmarks via comparing the performance with related baselines. In §3.4.3 we conduct the ablation study to demonstrate the effectiveness of our design.

---

<sup>1</sup><https://docs.dgl.ai/index.html>

<sup>2</sup><https://github.com/locuslab/qpth>

### 3.4.1 Training Self-Supervised Graph Neural Network

We follow the experimental setting in GCC [107] and use six graph datasets from NetRep [108] and SNAP [153, 8] covering academic graphs and social networks. In detail, we use Academia and two DBLP datasets for academic graphs, and Facebook, IMDB, and LiveJournal datasets for social networks. The statistics of the pre-training datasets are summarized in Table 6.

For our robust self-supervised method, we adopt a 5-layer 64-hidden-units GIN as the backbone model. We use one-hop neighborhood sampling with 5 neighbors in the first stage, and a random walk with restart probability 0.8 in the second stage. The query encoder is trained using Adam optimizer with learning rate of 0.001,  $\beta_1 = 0.5$ ,  $\beta_2 = 0.999$ , and  $\epsilon = 10^{-8}$ . We use the momentum contrast (MoCo) method [54] to update the key encoders. The mini-batch size is 32, and the dictionary size is 16384. The momentum is set to 0.9. The pre-train takes 50000 steps and the learning rate is decayed by  $\frac{1}{10}$  in step 10000 and step 30000. We include two versions of our approach, *proposed*<sup>Δ</sup> which omits the distributional robust consideration, and *proposed* which is the full approach.

### 3.4.2 Downstream Analysis

We consider three standard benchmark tasks for graph learning algorithms, *i.e.* node classification, graph classification, and top-k similarity search. Through the experiments, we use GCC [107] and its variants as the self-supervised baselines. Supervised baselines are detailed in the associated experiments to be discussed in the following.

#### 3.4.2.1 Node Classification

: In this task, we predict the unknown node labels in a partially labeled network. We adopt US-Airport and H-index as the benchmark datasets. US-Airport contains the airline activity levels among 1190 airports. H-index is an academic dataset indicating the distribution of the h-index of authors extracted from OAG [163]. We use the pre-trained query encoder to extract features and a logistic regression classifier to make the final prediction. Specifically,

we finetune the query encoder and the classifier for 90 epochs using a batch size of 128.

We consider ProNE [164], GraphWave [34], and Struct2Vec [41] as supervised baselines. The cardinality of subgraphs sets is 5. The results are presented in Table 7. For node classification, we find the pre-trained graph neural networks are superior to the supervised models. For example, our model outperforms the best supervised baseline, Struc2Vec, by up to 2.7. Compared to self-supervised baselines, our algorithm shows improvement with better stability for different datasets. The distributional robust consideration also improves the model performance significantly.

### 3.4.2.2 Graph Classification

: In this task, we predict the labels for graphs. We consider five datasets [151], including IMDB-Binary (IMDB-B), IMDB-Multi (IMDB-M), COLLAB, Reddit Binary (RDT-B), and Reddit-Multi5k (RDT-M). We finetune the pre-trained encoder and a classifier parallel to the node classification. The main difference is that we use the full graphs instead of  $r$ -ego networks.

We consider DGK [151], Graph2Vec [92], InfoGraph [118], DGCNN [166], and GIN [149] as supervised baselines. The results are presented in Table 8. Of note, our approach consistently outperforms the self-supervised baselines on most datasets. For COLLAB, our approach is competitive to the best-performing self-supervised baselines. Meanwhile, our method is also competitive to the supervised baselines. For example, our method has the same performance with the best-performing baseline, GIN, on IMDB-B, and is also very close on IMDB-M. In COLLAB, our model surpasses the best performing supervised baseline by a large margin.

### 3.4.2.3 Top-k Similarity Search

: Given two graphs, the top-k similarity search attempts to find the most similar vertices in one graph for the vertices in the other graphs. We adopt the co-author graphs [165] of K-I (KDD *v.s.* ICDM) S-C (SIGIR *v.s.* CIKM), and S-I (SIGMOD *v.s.* ICDE) and define the ground truth similarity as the common authors in both conferences. We use top-10 accuracy, *i.e.* HITS@10 as the performance measurements. Of note, this is an unsupervised task, and



we use the pre-trained model without finetuning. The baseline methods include random guess, RolX [55], Panther++ [165], and GraphWave [34]. The results are presented in Table 9. Compared to the in-place methods, such as Panther++, the self-supervised methods show competitive performances. Particularly, the proposed method show improvements compared to the state-of-the-art self-supervised methods.

### 3.4.3 Ablation Studies

#### 3.4.3.1 Model Robustness

: To evaluate the distributional robustness with the presence of distribution shift, we consider different sampling settings for building the structural embeddings, and the results are summarized in Fig. 9. We test five different sampling settings for the second stage: for neighborhood sampling with different neighbor size, we consider NS 4 and NS 5; for random walk with restart with different restart probability, we consider RWR 0.6, RWR 0.7 and RWR 0.8. The results are based on 10-fold validation accuracy on RDT-B. Different sampling settings will lead to different subgraph distributions. The results show that our method is insensitive to the choice of sampling settings.

#### 3.4.3.2 Cardinality of Embedding Set

: The first sampling stage in computing our structural embeddings is neighborhood sampling, and in the above experiments, we consider 5 neighbors within one-hop for each node of interest. In this section, we alter the value of  $C$ , the cardinality of the embedding set, to show its relationship with the model performance. The results are summarized in Fig. 10. The results show that with the growth of the neighbor size, the model performance first increases then stay stable. The cardinality used in this paper is chosen to balance the computational efficiency and the performance.

### 3.4.3.3 Computational Time

: Our approach outperforms related baselines in several representative graph-related tasks. A potential disadvantage is that our approach requires longer computational time compared to related pretraining graph methods. Table. 10 describes the pretraining time for the baseline GCC and our approach using different subgraph size (denoted by superscripts). It can be observed that our approach takes moderately longer training time than GCC [107], and the per-step time is positive correlated with the size of subgraphs.

## 3.5 Conclusion

In this paper, we propose a robust self-supervised graph neural network. We design structural embeddings to explicitly represent both the nodes of interest and their neighbors and utilize 1-Wasserstein distance to characterize network proximity. We formulate an asymptotically distributionally robust contrastive learning framework, which is reluctant to distributional shift. We exploit a differentiable optimization framework to compute the network proximity, which makes our model end-to-end trainable. The experimental results demonstrate that our approach outperforms state-of-the-art baselines in various downstream graph analyses and our design is effective in improving model robustness.

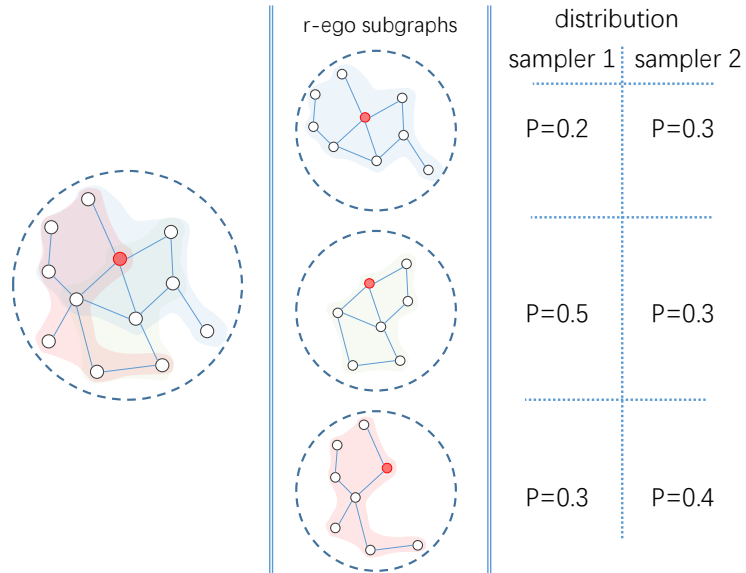


Figure 8: An example shows the distribution of  $r$ -ego subgraphs are determined by the sampling methods. The node of interest is colored in red, and three possible subgraphs are shaded in different colors (middle column). Under different sampling techniques (*e.g.* altering the backward jump probability in random walks), the distribution of subgraphs are presumably different (right column).

Table 6: The statistic for the pre-training datasets. Among these datasets, Academia, DBLP-SNAP, and DBLP-NetRep are academic datasets, and the rest are social networks.

Dataset	Academia	DBLP-SNAP	DBLP-NetRep	IMDB	Facebook	LiveJournal
# nodes	137969	317080	540486	896305	3097165	4843953
# edges	739384	2099732	30491458	7565894	47334788	85691368

Table 7: Node Classification Results.

Dataset	US-Airport	H-index
# nodes	1190	5000
# edges	13599	44020
ProNE	62.3	69.1
GraphWave	60.2	70.3
Struc2vec	66.2	–
GCC-E2E	65.3	77.7
GCC-MoCo	65.8	76.1
GCC-rand*	63.6	77.2
GCC-E2E*	68.4	78.8
GCC-MoCo*	66.5	80.9
Proposed <sup>Δ</sup>	68.1	80.7
Proposed	<b>68.9</b>	<b>81.2</b>

Table 8: Graph Classification Results.

Dataset	IMDB-B	IMDB-M	COLLAB	RDT-B	RDT-M
# graphs	1000	1500	5000	2000	5000
# classes	2	3	3	2	5
avg.# nodes	19.8	13.0	74.5	429.6	508.5
DGK	67.1	44.7	73.4	78.0	40.8
Graph2Vec	71.1	50.2	–	76.9	48.2
InfoGraph	73.3	50.1	–	83.1	53.5
GCC-E2E	71.5	49.3	74.8	86.9	53.1
GCC-MoCo	72.4	49.3	79.0	90.1	53.4
DGCNN	70.2	48.0	73.7	–	–
GIN	<b>75.6</b>	51.5	80.2	89.4	54.5
GCC-rand*	75.5	51.0	78.6.	87.9	52.0
GCC-E2E*	71.2	47.6	79.1	86.1	48.6
GCC-MoCo*	73.3	50.5	81.1	88.0	53.2
Proposed <sup>Δ</sup>	74.9	53.0	80.7	89.9	55.4
Proposed	<b>75.6</b>	<b>53.1</b>	<b>81.2</b>	<b>90.4</b>	<b>55.9</b>

Table 9: Top-k Similarity Search ( $k = 40$ ). Best-performing self-supervised methods are bold faced. Best performing non-self-supervised methods are denoted by \*.

Dataset	K-I	S-C	S-I
—V—	2607	3548	2559
—E—	4774	7076	6668
# ground truth	697	874	898
Random	0.0566	0.0447	0.0521
RoIX	0.1288	0.0984	0.1309
Panther++	0.1558	0.1185*	0.1320
GraphWave	0.1693*	0.0995	0.1470*
GCC-E2E	0.1564	<b>0.1247</b>	0.1336
GCC-MoCo	0.1521	0.1178	0.1425
Proposed	<b>0.1588</b>	0.1191	<b>0.1439</b>

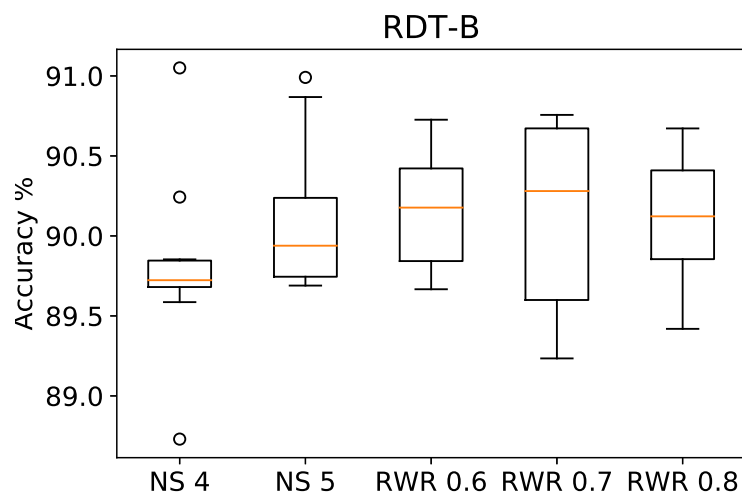


Figure 9: The model performance under different subgraph distributions. We test five different sampling settings for the second stage: for neighborhood sampling with different neighbor size, we consider NS 4 and NS 5; for random walk with restart with different restart probability, we consider RWR 0.6, RWR 0.7 and RWR 0.8. The results are based on 10-fold validation accuracy on RDT-B. Our model performs similar under different settings, which indicates our method is robust to distribution shift.

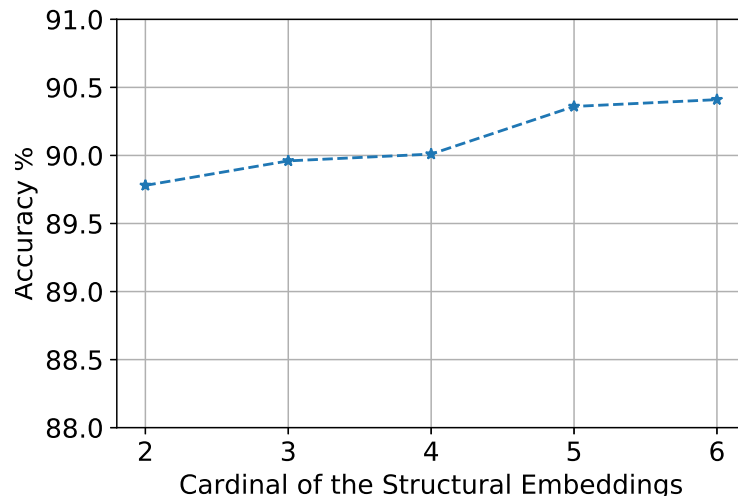


Figure 10: Model performance *v.s.* sampling size.

Table 10: The comparison of per-step time for the baseline and our method using different subgraph size (denoted by superscripts).

Method	Time (ms)
GCC MoCo	4.31
Proposed <sup>3</sup>	5.87
Proposed <sup>4</sup>	5.93
Proposed <sup>5</sup>	6.16



## 4.0 Training Graph Neural Network Faster using Stale Gradients

### 4.1 Motivation

Convolutional Neural Networks (CNN) [72, 73] have been successfully used in various machine learning tasks with grid-structural data, including machine learning and natural language processing [67, 89, 53, 62, 61]. Inspired by the same principle, Graph Convolutional Network (GCN) [17, 30, 37, 94] and its variants [131, 64, 50] are proposed to handle arbitrarily structural data, and have achieved state-of-the-art results in many applications, for example graph embedding and node classification [51, 133, 124, 49, 65, 102, 112]. Compared to neural networks, GCN can learn better node expressions based on both the nodes and their neighbors. Particularly, the association between distant nodes can be captured by applying high-order graph kernel or stacking multiple layers.

Despite the benefits of GCN, a severe problem lies in the training on large dataset [24, 23, 58]. Efficient training methods, such as stochastic gradient descent, can process data on affordable batches. However, when training GCN a similar procedure is obstructed by the “batch explosion”, as illustrated in Fig. 11. The inference of a single node requires the induced sub-graph constructed on the receptive fields. With multiple stacked layers, the receptive fields are yielded by recursively crawling along neighbored nodes in the graphs. Eventually, a small batch will grow explosively into a large vertices set, and for dense graphs or deep structures, the receptive fields may even take over the whole graphs. This phenomenon will result in the annoyingly high consumption of memory and time in stochastic methods.

There have been some attempts to accelerate the training of GCN. An initial study is GraphSage [50], in which a scheme of node-wise neighbor sampling (NS) is developed. Instead of considering all neighbors,  $D^l$  neighbors are sampled at the  $l$ -th layer to reduce the receptive field size. The hidden units are estimated based on the sampled sub-graphs. [24] further proposed to use layer-wise importance sampling (IS) to decrease the sample variance and attain a better estimation. Adaptive sampling [58] is another layer-wise sampling method, which is shown to yield more accurate results by learning a sampling function. Skip links are

also used to promote the message passing between distant nodes. To improve the estimator, [23] modified the model using control variable method, through storing historical states and approximated the updating accordingly. To sum up, all these methods focus on reducing the estimation variance.

In this paper we propose a faster training method for GCN from a different perspective. A close investigation on the “batch explosion” will reveal the fact that, the bottleneck is introduced by both the network depth and the sampling behavior. Instead of improving the sampling, we focus on “*reducing*” the depth. However, directly using shallow networks is opposed to the principle of exploiting deep structures [39, 116, 121], thus not an immediate solution to our purpose. To overcome this difficulty, we propose to divide the naive GCN model into sequential sub-models during training. In summary, our major **contributions** are given as follows:

- 1) We re-formulate the original model optimization problem into a constraint problem. Moreover, we propose to use *gradient flashback* to break the sampling and updating dependency between sub-models, which enables a simultaneously training.
- 2) We provide a theoretical analysis to established the convergence for non-convex problems under certain conditions, and provide a discussion on the variance of the sampled neighbor sets.
- 3) The experiments demonstrate that the proposed method can greatly reduce the running time per epoch compared to naive stochastic methods, and achieve consistent results in both convergence rate and generalization ability. We also show that the proposed method is compatible with various network designs and sampling schemes.

#### 4.1.1 Organization and Notations

The rest of this paper is organized as follow: §4.2 presents the necessary background and related work; §4.3 described the proposed method; §4.4 provides the theoretical results on convergence analysis; §4.6 shows the experiment results; §4.7 concludes the paper.

Throughout the paper,  $\|\cdot\|$  denotes the vector  $\ell_2$  norm and the matrix spectral norm, respectively. If not specified, matrices are denoted using bold upper case letters, vectors are

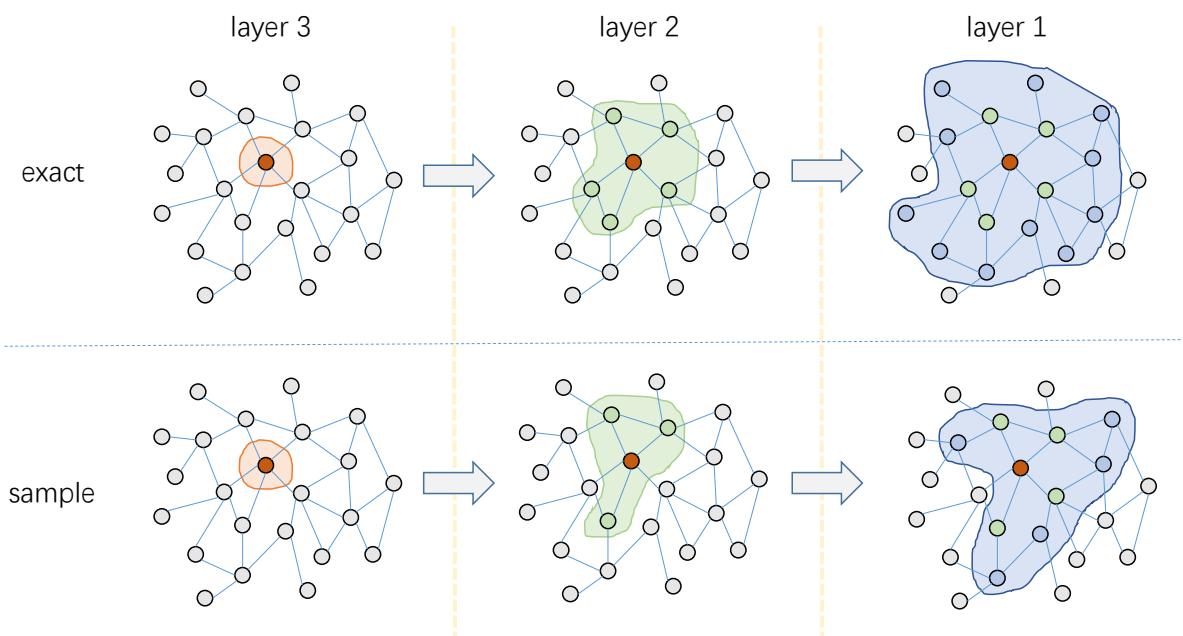


Figure 11: The growing of the receptive fields layer-wise. Top line: the growing for exact method; bottom line: the growing for sampled methods. Involved nodes in each nodes are labelled with different colors. Clearly, deep network causes the size of involved nodes in bottom layers exploded, particularly for exact method.

in bold lower case letters, and scalars are in plain letters.

## 4.2 Preliminary and Related Work

### 4.2.1 Preliminary

A graph can be represented as  $\{V, E, \mathbf{W}\}$ , with  $V = \{v_1, v_2, \dots, v_n\}$  the set of  $n$  vertices,  $E \subseteq V \times V$  the set of  $m$  edges, and  $\mathbf{W}$  the weighted adjacency matrix of the graph. The adjacency matrix  $\mathbf{W} \in \mathbb{R}^{n \times n}$ , and  $W_{ij}$  encodes the relation between  $v_i$  and  $v_j$  if there is an edge, and 0 otherwise. The degree matrix of a graph can be expressed with diagonal matrix  $\mathbf{D} \in \mathbb{R}^{n \times n}$ , with the diagonal entries  $d_{ii} = \sum_j w_{i,j}$ , representing the degree of node  $i$ . Graph Laplacian is an essential operation in spectral graph analysis, typically defined as  $\mathbf{L}_c = \mathbf{D} - \mathbf{W}$  in combinatorial form and  $\mathbf{L}_n = \mathbf{I}_n - \mathbf{D}^{-1/2} \mathbf{W} \mathbf{D}^{-1/2}$  in normalized form, with  $\mathbf{I}_n$  a identity matrix. GCN is designed to analysis the signals on nodes, with a given graph structure. As the name indicates, GCN is an extension of CNN. Convolution on 2-D matrices, for example images, is well defined, serving as the basic building block in CNN. Graph convolution, however, is difficult to perform directly in spatial domains, due to the irregularity of neighbors of distinct nodes. One strategy is to conduct the operation in frequency domain using graph Laplacian, which is feasible according to the convolution theorem [84]. By definition, Laplacian matrix is positive semi definite, such that the eigenvalue decomposition exists,  $\mathbf{L} = \mathbf{U}^\top \mathbf{\Lambda} \mathbf{U}$ , and  $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_{n-1}]$  specifies a Fourier basis. The graph Fourier transform [115] is then defined as  $\hat{\mathbf{x}} = \mathbf{U}^\top \mathbf{x}$ , with  $\mathbf{x} \in \mathcal{R}^n$  the signal, and the inverse transform  $\mathbf{x} = \mathbf{U} \hat{\mathbf{x}}$ . The spectral representation of node signals,  $\hat{\mathbf{x}}$ , allows the fundamental filtering operation for graphs.

One potential problem here is that in spectral domain, the filter is not naturally localized. Polynomial parametrization for localized filters [30] is proposed to tackle this challenge, through learning the coefficients  $\Theta_K$  of a  $K$ -order Chebyshev polynomial. In detail, the filter is defined as  $g_\theta(\mathbf{\Lambda}) = \sum_{k=0}^{K-1} \theta_k T_k(\mathbf{\Lambda})$ , and the graph convolution is defined as  $y = \mathbf{U} g_\theta(\mathbf{\Lambda}) \mathbf{U}^\top \mathbf{x}$ , with  $y$  the filtered signal,  $\theta_k$  the trainable parameters and  $T_k(\mathbf{\Lambda})$  the polynomials. By the

stable recurrence relation of Chebyshev polynomial, the costly multiplication in Fourier transform can also be reduced. Parallel to CNN, pooling operation in graph settings is accomplished by the graph coarsening [33] procedure.

The above method is suitable for graph classification, which is extended to node classification later [64]. By truncating the Chebyshev polynomial to only first order, GCN model can be reformulated similar to multi-layer perceptron, with each layer defined as  $y = \sigma(\tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2} \mathbf{x} \Theta)$ , with  $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$ ,  $\tilde{\mathbf{D}}$  defined as aforementioned  $\mathbf{D}$ ,  $\Theta$  the trainable parameters, and  $\sigma(\cdot)$  an activation function. In the proposed method we will adopt the first order approximation version of GCN as the first stage to obtain embeddings for a graph, then use a MLP in the second stage for the regression task.

#### 4.2.2 Related Work

The fast training of GCN usually exploits stochastic methods with back-propagation [110]. Previous studies mainly focus on reducing the receptive fields via sampling techniques. Based on the sampling manners, these methods can be separated into two categories, node-wise sampling and layer-wise sampling. Neighbor sampling (NS) proposed by [50] is a typical node-wise sampling method. For each layer, NS randomly samples  $D^l$  neighbors for each node and estimator the value as,

$$(\mathbf{P}\mathbf{H}^l)_u \approx \frac{n(u)}{D^l} \sum_{v \in \hat{n}_u} P_{uv} \mathbf{h}_v^l, \quad (4.1)$$

here  $\mathbf{P} = \tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2}$ ,  $\hat{n}_u$  is a subset of  $D^l$  random neighbors of node  $u$ . Instead of sampling neighbors for each node, layer-wise methods directly sample the receptive field for each layer altogether. Given a sampling distribution  $q(v)$ , the estimation is,

$$(\mathbf{P}\mathbf{H}^l)_u \approx \frac{V}{S} \sum_{v_s \sim q(v)} P_{uv} \mathbf{h}_{v_s}^l / q(v_s), \quad (4.2)$$

here  $V$  and  $S$  are the node number and sample size, respectively. Importance sampling (IS) [24] and adaptive sampling (AS) [58] are two notable layer-wise methods. IS uses importance distribution to reduce the estimation variance. AS learns a sample distribution conditioned on the higher layer, to capture the node relations.

However, there are some drawbacks in these methods. For node-wise sampling, the receptive field is still growing by  $\mathcal{O}(\prod_l D_l)$ , which is not applicable for dense graphs. Meanwhile, node-wise methods have a similar problem in the growth. Moreover, in practise node-wise methods can sample no neighbors for some nodes, which lead to meaningless zero activation values.

### 4.3 Gradient Flashback Method

In this section, we propose a novel gradient flashback method for training large-scale GCN.

First, we review the naive GCN model [64] with  $L$  layers, which is written as,

$$\mathbf{h}_i^{l+1} = \sigma(\mathbf{P}\mathbf{h}_i^l\mathbf{W}^l), \quad l = 1, 2, \dots, L, \quad (4.3)$$

here  $\mathbf{h}_i^l$  is the inputs of sample  $i$  to  $l$ -th layer, and  $\mathbf{h}_i^{l+1}$  is the outputs, and  $\mathbf{P}$  is the graph-structure matrix,  $\mathbf{W}^l$  is the trainable weights of  $l$ -th layer, and  $\sigma(\cdot)$  is activation function, and  $L$  is the number of layers.  $\mathbf{h}_i^1$  is defined as sample  $\mathbf{x}_i$ , and the outputs of the last layer  $\mathbf{h}_i^{L+1}$  is defined as outputs  $\hat{y}_i$ . Then training this model can be described the following problem

$$\min_W \mathcal{L}(\mathbf{X}, \mathbf{Y}, \mathbf{W}), \quad (4.4)$$

where  $W = [\mathbf{W}^1, \mathbf{W}^2, \dots, \mathbf{W}^L]$  is the collection of trainable parameters,  $\mathbf{X}$  are the inputs in matrix form,  $\mathbf{Y}$  are the corresponding targets, and  $\mathcal{L}(\cdot)$  is the loss function.

Recently, to training large-scale deep neural networks, the stochastic methods are widely used, which include the stochastic gradient descent (SGD [109]) and its variants such as Adam [63]. Similarly, these stochastic methods such as SGD can also be used to train the GCN, but an additional layer-wise sampling is also required. In fact, the updating of shallow layers are dependent on deeper layers in both gradient computing and sampling, which is highly inefficient. Thus, we propose a fast gradient flashback method to train the large-scale GCN. Specifically, the proposed method partitions the naive GCN model into a sequence of sub-models, and accomplishes the sampling and updating using historical error gradients,

which breaks the dependency and accelerates the training. The sequential sub-models can be written as,

$$\mathbf{h}_{i_{k+1}} = f_k(\mathbf{h}_{i_k}, \mathbf{W}_k), \quad k = 1, 2, \dots, K, \quad (4.5)$$

where  $f_k(\cdot)$  is the  $k$ -th sub-model defined as in (4.3),  $\mathbf{h}_{i_k}$  and  $\mathbf{h}_{i_{k+1}}$  the inputs and outputs, respectively, and  $\mathbf{W}_k$  is the wrapped trainable parameter of  $f_k(\cdot)$ .

With the modified GCN model, we can transform problem (4.4) into a constrained problem as follows:

$$\begin{aligned} \min_W \mathcal{L}_K(\mathbf{H}_K, \mathbf{W}_K), \\ \text{s.t. } \mathbf{H}_1 = \mathbf{X}, \mathbf{H}_{k+1} = f_k(\mathbf{H}_k, \mathbf{W}_k), \quad k = 1, \dots, K-1, \end{aligned} \quad (4.6)$$

To solve the above problem (4.6), we use the following iterative rule to update parameters:

$$\begin{cases} \mathbf{W}_k^{t+1} = \mathbf{W}_k^t - \gamma \frac{\partial f_k(\mathbf{H}_k^t, \mathbf{W}_k^t)}{\partial \mathbf{W}_k^t} g_k^{t-K+k}, \\ \mathbf{H}_{k+1}^{t+1} = f_k(\mathbf{H}_k, \mathbf{W}_k^{t+1}), \quad k = 1, \dots, K-1 \\ \mathbf{W}_K^{t+1} = \mathbf{W}_K^t - \gamma \frac{\partial \mathcal{L}_K(\mathbf{H}_K^t, \mathbf{W}_K^t)}{\partial \mathbf{W}_K^t}, \end{cases} \quad (4.7)$$

with

$$g_k^{t-k} = \frac{\partial f_{k+1}(\mathbf{H}_{k+1}^{t-K+k}, \mathbf{W}_{k+1}^{t-K+k})}{\partial \mathbf{H}_{k+1}^{t-K+k}},$$

where  $t - K + k \leq 0$ . In fact, this iterative optimization method can be viewed as a variant of stochastic gradient descent method. The main modification is that, for each sub-model, we update the outputs in real-time as well as storing residual error gradients and sampled nodes in the forward step, and train the sub-model with stale error gradients in the back-propagation. For each batch, different sub-models can be trained simultaneously, instead of waiting for the sub-graph sampling and error gradients in current batch. The error gradients are computed in  $t$ -th step and flash back after several steps for training, which is the reason we name the method *gradient flashback*. Intuitively the gradient flashback will become close to real-time gradients, and in next section, a theoretical analysis will be given.

Finally, the proposed training algorithm is summarized in Algo. 2. Then we combined the proposed method with neighbor sampling, and the sampling procedure is summarized in Algo. 3.

---

**Algorithm 2** Fast training GCN using gradient flashback

---

**Input:** Graph  $G$ , Features  $\mathbf{X}$ , Labels  $\mathbf{Y}$ , Epoch Limits  $T$ , Sub-Models numbers  $K$

**Output:** Model Parameter  $\mathbf{W}$

```
1 initialization;
2 compute  $\mathbf{P}$ ;
3 construct  $K$  sub-models, each with parameters  $\mathbf{W}_k$ ;
4 while  $epoch < T$  do
5   for  $k \leq K$  do
6     require hidden states  $\overline{\mathbf{H}}_{k-1}$  as inputs to sub-model  $k$ ,  $\overline{\mathbf{H}}_0 = \mathbf{X}$ ;
7     forward compute  $H_k$ , update stale hidden states  $\overline{\mathbf{H}}_k$ ;
8   end
9   while  $epoch \text{ not end}$  do
10    shuffle data;
11    for  $k \leq K$  do
12      if  $epoch \geq K-k$  then
13        pass
14      else
15        obtain batch data and approximate  $\hat{\mathbf{P}}$  following Algo. 3;
16        update  $\mathbf{W}_k$  w.r.t. (4.7), compute  $g_{k-1}$ ;
17        update stale gradients  $\overline{g}_{k-1}$  and stale batch  $\overline{V}_{k-1}$  with  $g_{k-1}$  and  $V_{k-1}$ 
18        respectively;
19      end
20    end
21 end
```

---



---

**Algorithm 3** Fetching batch data

---

**Input:** Graph matrix  $\mathbf{P}$ , Current epoch  $t$ , Sub-Model index  $k$

**Output:**

```
1 initialization;
2 if  $k$  is  $K$  then
3   | obtain a new batch  $\mathbf{V}_K$  from all vertices;
4 else
5   | fetch historical batch  $\overline{\mathbf{V}}_k$ ;
6 end
7 for  $l \leq l^k$  do
8   | for  $u \in V_k$  do
9     | sample  $\hat{n}_u$  from neighbor set  $n_u$  of vertex  $u$ ;
10    | for  $v \in \hat{n}_u$  do
11      | update  $\hat{P}_{uv} = \|n(u)\|/\|\hat{n}_u\|P_{uv}$ ;
12    | end
13  | end
14 end
15 collect all sampled vertices to construct  $V_{k-1}$ ;
16 return  $V_{k-1}, \overline{\mathbf{P}}_k^l$ ;
```

---

## 4.4 Convergence Analysis

In this section, we demonstrate that the proposed method is guaranteed to converge to a critical point for non-convex problems given certain conditions, and the convergence rate is sub-linear.

We begin with introducing the common-used assumptions in SGD, which are also adopted throughout our analysis,

**Assumption 1.** *Throughout the paper we make the same assumptions ensuring the convergence of SGD on the candidate problem [16],*

1. **(Smoothness)** *The objective function  $\mathcal{L} : \mathbb{R}^d \rightarrow \mathbb{R}$  is continuously differentiable and the gradient function of  $\mathcal{L}$ , namely,  $\nabla \mathcal{L} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ , is Lipschitz continuous with Lipschitz constant  $L > 0$ , i.e.,*

$$\|\nabla \mathcal{L}(w) - \nabla \mathcal{L}(\bar{w})\|_2 \leq L \|w - \bar{w}\|, \quad w, \bar{w} \in \mathbb{R}^d. \quad (4.8)$$

2. **(Moment limits)** *The objective function and SGD satisfy the following:*

- a. *The sequence of iterates  $\{w^t\}$  is contained in an open set over which  $\mathcal{L}$  is bounded below by a scalar  $L_{inf}$ .*
- b. *The updating is descending in expectation, namely there exist  $\mu$ ,*

$$\langle \nabla \mathcal{L}(w_t), \mathbb{E} \rangle \geq \mu \|\nabla \mathcal{L}(w_t)\|_2^2. \quad (4.9)$$

- c. *The second moment of the gradients of objective is upper bounded, that is, there exist  $M \geq 0$ , such that,*

$$\mathbb{E}[\|\nabla \mathcal{L}(w^t)\|_2^2] \leq M, \quad \text{for } \mathbf{H}, W. \quad (4.10)$$

To bridge the gap between the proposed method and SGD, we decompose the gradient flashback into immediate gradient plus disturbing terms. Thus we introduce another assumption to constraint the disturbing term.

**Assumption 2.** *The following assumptions evaluate the gradient flashback method,*

1. **(Bounded Sub-model Gain)** The spectral norm of the gradients of sub-models are upper-bounded, that is, there exist a constant  $\phi_1 \geq 0$  such that,

$$\left\| \frac{\partial f_k(\mathbf{H}_k, \mathbf{W}_k)}{\partial \mathbf{H}_k} \right\| \leq \phi_1, \quad k = 1, 2, \dots, K-1, \quad (4.11)$$

2. **(Bounded Objective Gain)** The spectral norm of the second order gradients of the objective regarding the parameters are upper bounded, that is, there exist a constant  $\phi_2 \geq 0$  such that,

$$\begin{aligned} \left\| \frac{\partial^2 \mathcal{L}_K(\mathbf{H}_K, \mathbf{W}_K)}{\partial \mathbf{H}_K \partial \mathbf{W}_K} \right\| &< \phi_2, \\ \left\| \frac{\partial^2 \mathcal{L}_K(\mathbf{H}_K, \mathbf{W}_K)}{\partial \mathbf{H}_K^2} \right\| &< \phi_2, \end{aligned} \quad (4.12)$$

Based on the above assumptions, we study the convergence rate of the proposed method in the following.

**Theorem 2.** In Algorithm 2, given an appropriate  $\gamma$ , there exist constants  $M_k, \eta_k^* \in \mathbb{R}^+$ , for  $k \in \mathbb{N}^+$ , we have

$$\frac{1}{T} \sum_{t=0}^T \mathbb{E} \left[ \|\nabla \mathcal{L}_2^t\|_2^2 \right] \leq \frac{\mathcal{L}_2^0 - \mathcal{L}_2^*}{T} + \frac{L\gamma^2}{2} \sum_{k=1}^K M_k \eta_k^*, \quad (4.13)$$

where  $\mathcal{L}_2^*$  is the optimal value of the objective,

**Remark 2.** Theorem 2 shows that under some mild conditions, our algorithm has a convergence rate of  $O(1/T)$ , which is the same with naive stochastic methods. By simultaneously training multiple shallow sub-models, we can significantly reduce the time complexity per epoch and accelerate stochastic training. We also include an analysis on the estimation of sampled sub-graphs, which shows the necessity of batch normalization.

**Theorem 3.** Given the neighbor set  $n_u$  of vertex  $u$ ,  $\hat{n}_u$  contains  $|\hat{n}_u|$  nodes sampled using distribution  $\{q_v\}$  without replacement, and the estimator is unbiased. Assume

$$\mathbb{E} \left[ \left( \hat{P}_{uv_i} - P_{uv_i} \right) \left( \hat{P}_{uv_j} - P_{uv_j} \right) \mathbf{x}_{v_i} \mathbf{x}_{v_j} \right] = 0, \quad (4.14)$$

then there exists a constant  $c_s > 0$ , such that the second centre moments of the approximation satisfies,

$$\text{Var}_u(\hat{\mathbf{P}}_u \mathbf{x}_u) \leq c_s \sum_{v \in n_u} (P_{uv} \mathbf{x}_v)^2. \quad (4.15)$$

**Remark 3.** *Theorem 3 shows that compared to the exact method, the estimator admits a moderate variance increase. This results provide some insights on the stability of the training. For neighbor sampling, the equality holds. Particularly, the above results show that through normalization, the variance can be greatly reduced regardless of the sampling methods. Although the naive GCN [64] does not exploit batch normalization, this result can help explain the performance improvement of successive works utilizing this techniques [23].*

## 4.5 Appendix

In this section, we at detail provide the proofs of the above theorems.

### 4.5.1 Proof of Theorem 1

*Proof.* Without loss of generality, we first consider a naive GCN split into two sub-models, namely  $K = 2$ . This problem can be described as follows:

$$\min_{\mathbf{W}_1, \mathbf{W}_2} \mathcal{L}_2(\mathbf{H}, \mathbf{W}_2), \quad \text{s.t. } \mathbf{H} = f_1(\mathbf{X}, \mathbf{W}_1), \quad (4.16)$$

To solve the above problem (4.16), we use an updating rule as follows:

$$\begin{cases} \mathbf{W}_1^{t+1} = \mathbf{W}_1^t - \gamma \frac{\partial f_1(\mathbf{X}, \mathbf{W}_1^t)}{\partial \mathbf{W}_1^t} \frac{\partial \mathcal{L}_2(\mathbf{H}^{t-1}, \mathbf{W}_2^{t-1})}{\partial \mathbf{H}^{t-1}}, \\ \mathbf{W}_2^{t+1} = \mathbf{W}_2^t - \gamma \frac{\partial \mathcal{L}_2(\mathbf{H}^t, \mathbf{W}_2^t)}{\partial \mathbf{W}_2^t}, \\ \mathbf{H}^{t+1} = f_1(\mathbf{X}, \mathbf{W}_1^{t+1}), \end{cases} \quad (4.17)$$

For simplicity, let  $\mathcal{L}_2^{t_1, t_2} \doteq \mathcal{L}_2(\mathbf{H}^{t_1}, \mathbf{W}_2^{t_2})$ , and  $\mathcal{L}_2^t \doteq \mathcal{L}_2(\mathbf{H}^t, \mathbf{W}_2^t)$ . Following SGD, we have

$$\mathcal{L}_2^{t+1} - \mathcal{L}_2^t \leq -\langle \nabla \mathcal{L}_2^t, \mathbf{W}^{t+1} - \mathbf{W}^t \rangle + \frac{L}{2} \|\mathbf{W}^{t+1} - \mathbf{W}^t\|^2, \quad (4.18)$$

where  $\mathbf{W}^t \doteq [\mathbf{W}_1^t, \mathbf{W}_2^t]$ , and

$$\|\mathbf{W}^{t+1} - \mathbf{W}^t\|^2 = \|\mathbf{W}_1^{t+1} - \mathbf{W}_1^t\|_2^2 + \|\mathbf{W}_2^{t+1} - \mathbf{W}_2^t\|^2. \quad (4.19)$$

Here  $\mathbf{W}_2$  term is identical to SGD settings. For  $\mathbf{W}_1$  term, we use the gradient flashback to update as follows:

$$\begin{aligned}
\gamma^{-2} \|\mathbf{W}_1^{t+1} - \mathbf{W}_1^t\|^2 &= \left\| \frac{\partial \mathbf{H}^t}{\partial \mathbf{W}_1^t} \frac{\partial \mathcal{L}_2^{t-1}}{\partial \mathbf{H}^{t-1}} \right\|^2 = \left\| \frac{\partial \mathbf{H}^t}{\partial \mathbf{W}_1^t} \left[ \frac{\partial \mathcal{L}_2^t}{\partial \mathbf{H}^t} + \left( \frac{\partial \mathcal{L}_2^{t-1}}{\partial \mathbf{H}^{t-1}} - \frac{\partial \mathcal{L}_2^t}{\partial \mathbf{H}^t} \right) \right] \right\|^2 \\
&= \left\| \frac{\partial \mathbf{H}^t}{\partial \mathbf{W}_1^t} \left[ \frac{\partial \mathcal{L}_2^t}{\partial \mathbf{H}^t} + \left( \frac{\partial \mathcal{L}_2^{t,t-1}}{\partial \mathbf{H}^t} - \frac{\partial \mathcal{L}_2^t}{\partial \mathbf{H}^t} \right) + \left( \frac{\partial \mathcal{L}_2^{t-1}}{\partial \mathbf{H}^{t-1}} - \frac{\partial \mathcal{L}_2^{t,t-1}}{\partial \mathbf{H}^t} \right) \right] \right\|^2 \\
&\leq \underbrace{\left\| \frac{\partial \mathbf{H}^t}{\partial \mathbf{W}_1^t} \frac{\partial \mathcal{L}_2^t}{\partial \mathbf{H}^t} \right\|^2}_{T_1} + \underbrace{\left\| \frac{\partial \mathbf{H}^t}{\partial \mathbf{W}_1^t} \left( \frac{\partial \mathcal{L}_2^{t,t-1}}{\partial \mathbf{H}^t} - \frac{\partial \mathcal{L}_2^t}{\partial \mathbf{H}^t} \right) \right\|^2 + \left\| \frac{\partial \mathbf{H}^t}{\partial \mathbf{W}_1^t} \left( \frac{\partial \mathcal{L}_2^{t-1}}{\partial \mathbf{H}^{t-1}} - \frac{\partial \mathcal{L}_2^{t,t-1}}{\partial \mathbf{H}^t} \right) \right\|^2}_{T_2},
\end{aligned} \tag{4.20}$$

where the term  $T_1$  is the same as the classic gradient descent method, and the term  $T_2$  is the disturbance introduced by the gradient flashback. Using Assumption. 2, the second term can be bounded,

$$\begin{aligned}
\left\| \frac{\partial \mathbf{H}^t}{\partial \mathbf{W}_1^t} \left( \frac{\partial \mathcal{L}_2^{t,t-1}}{\partial \mathbf{H}^t} - \frac{\partial \mathcal{L}_2^t}{\partial \mathbf{H}^t} \right) \right\|^2 &= \left\| \frac{\partial \mathbf{H}^t}{\partial \mathbf{W}_1^t} \int_0^1 \frac{\partial^2 \mathcal{L}_2(\mathbf{H}^t, \mathbf{W}_2^{t-1} + \delta(\mathbf{W}_2^t - \mathbf{W}_2^{t-1}))}{\partial \mathbf{H}^t \partial \delta} d\delta \right\|^2 \\
&\leq \left\| \frac{\partial \mathbf{H}^t}{\partial \mathbf{W}_1^t} \int_0^1 \frac{\partial^2 \mathcal{L}_2(\mathbf{H}^t, \mathbf{W}^\delta)}{\partial \mathbf{H}^t \partial \mathbf{W}^\delta} (\mathbf{W}_2^t - \mathbf{W}_2^{t-1})^T d\delta \right\|^2 \\
&\leq \int_0^1 \left\| \frac{\partial \mathbf{H}^t}{\partial \mathbf{W}_1^t} \right\|^2 \left\| \frac{\partial^2 \mathcal{L}_2(\mathbf{H}^t, \mathbf{W}^\delta)}{\partial \mathbf{H}^t \partial \mathbf{W}^\delta} \right\|^2 \|\mathbf{W}_2^t - \mathbf{W}_2^{t-1}\|^2 d\delta \leq \frac{\eta_1}{\gamma^2} \|\mathbf{W}_2^t - \mathbf{W}_2^{t-1}\|^2,
\end{aligned} \tag{4.21}$$

$$\tag{4.22}$$

where  $\eta_1 = \phi_1^2 \phi_2^2$  is the deduced bound coefficient. Similarly, the third term can be bounded

$$\left\| \frac{\partial \mathbf{H}^t}{\partial \mathbf{W}_1^t} \left( \frac{\partial \mathcal{L}_2^{t-1}}{\partial \mathbf{H}^{t-1}} - \frac{\partial \mathcal{L}_2^{t,t-1}}{\partial \mathbf{H}^t} \right) \right\|^2 \leq \eta_t \|\mathbf{H}^t - \mathbf{H}^{t-1}\|_2^2 \leq \frac{\eta_2}{\gamma^2} \|\mathbf{W}_1^t - \mathbf{W}_1^{t-1}\|^2,$$

where  $\eta_t$  and  $\eta_2$  are the deduced bound coefficients similar to  $\eta_1$ . The two inequalities follow by twice linear integration, also using Assumption. 2. Without loss of generality, we assume

$\eta_2 < 1$ . This can be ensured by simply shrinking the hidden units for early sub-models and amplifying the results back in the last sub-model. Take the expectation of both size of (4.18),

$$\begin{aligned} \mathbb{E}[\mathcal{L}_2^{t+1}] - \mathcal{L}_2^t &\leq -\langle \nabla \mathcal{L}_2^t, \mathbb{E}[\mathbf{W}^{t+1} - \mathbf{W}^t] \rangle + \frac{L}{2} \mathbb{E} \left[ \|\mathbf{W}_2^{t+1} - \mathbf{W}_2^t\|^2 \right] + \frac{L\gamma^2}{2} \mathbb{E} \left[ \left\| \frac{\partial \mathbf{H}^t}{\partial \mathbf{W}_1^t} \frac{\partial \mathcal{L}_2^t}{\partial \mathbf{H}^t} \right\|^2 \right] \\ &\quad + \frac{L\eta_1}{2} \mathbb{E} \left[ \|\mathbf{W}_2^t - \mathbf{W}_2^{t-1}\|^2 \right] + \frac{L\eta_2}{2} \mathbb{E} \left[ \|\mathbf{W}_1^t - \mathbf{W}_1^{t-1}\|^2 \right] \\ &\leq -\mu \|\nabla \mathcal{L}_2^t\|^2 + \frac{L\gamma^2}{2} \mathbb{E} \left[ \left\| \frac{\partial \mathbf{H}^t}{\partial \mathbf{W}_1^t} \frac{\partial \mathcal{L}_2^t}{\partial \mathbf{H}^t} \right\|^2 \right] + \frac{L\gamma^2}{2} \mathbb{E} \left[ \left\| \frac{\partial \mathcal{L}_2^t}{\partial \mathbf{W}_2^t} \right\|^2 \right] + \end{aligned} \quad (4.23)$$

$$\frac{L\eta_2}{2} \mathbb{E} \left[ \|\mathbf{W}_1^t - \mathbf{W}_1^{t-1}\|^2 \right] + \frac{L\gamma^2\eta_1}{2} \mathbb{E} \left[ \left\| \frac{\partial \mathcal{L}_2^{t-1}}{\partial \mathbf{W}_2^{t-1}} \right\|^2 \right]. \quad (4.24)$$

Then summing over  $t$ , we have

$$\begin{aligned} \mathbb{E}[\mathcal{L}_2^T] - \mathcal{L}_2^0 &\leq -\mu \sum_{t=0}^{T-1} \mathbb{E} \left[ \|\nabla \mathcal{L}_2^t\|_2^2 \right] + \frac{L\gamma^2}{2} \sum_{t=0}^{T-1} \mathbb{E} \left[ \left\| \frac{\partial \mathcal{L}_2^t}{\partial \mathbf{W}_2^t} \right\|^2 \right] + \frac{L\gamma^2\eta_1}{2} \sum_{t=0}^{T-1} \mathbb{E} \left[ \left\| \frac{\partial \mathcal{L}_2^{t-1}}{\partial \mathbf{W}_2^{t-1}} \right\|^2 \right] \\ &\quad + \frac{L\gamma^2}{2} \sum_{t=0}^{T-1} \mathbb{E} \left( \left\| \frac{\partial \mathbf{H}^t}{\partial \mathbf{W}_1^t} \frac{\partial \mathcal{L}_2^t}{\partial \mathbf{H}^t} \right\|^2 \right) + \frac{L\eta_2}{2} \sum_{t=0}^{T-1} \mathbb{E} \left[ \|\mathbf{W}_1^t - \mathbf{W}_1^{t-1}\|^2 \right] \\ &\leq -\mu \sum_{t=0}^T \mathbb{E} \left[ \|\nabla \mathcal{L}_2^t\|_2^2 \right] + \frac{L\gamma^2 M_1 T}{2(1-\eta_2)} + \frac{L\gamma^2 M_2 T (\eta_1 - \eta_2 + 1)}{2(1-\eta_2)}, \end{aligned} \quad (4.25)$$

where the last inequality follows by  $\eta_2 < 1$ , whose the last term is a shrunken version  $\sum_{t=0}^{T-1} \mathbb{E} \left[ \|\mathbf{W}_1^{t+1} - \mathbf{W}_1^t\|_2^2 \right]$ . Here  $M_1$  and  $M_2$  are the upper bounds of the gradient second moment of  $W_1$  and  $W_2$  accordingly. Since  $\mathcal{L}_2^*$  is the optimal value, we have  $\mathcal{L}_2^0 - \mathbb{E}(\mathcal{L}_2^T) \leq \mathcal{L}_2^0 - \mathcal{L}_2^*$ . Then we obtain

$$\frac{1}{T} \sum_{t=0}^T \mathbb{E} \left( \|\nabla \mathcal{L}_2^t\|_2^2 \right) \leq \frac{\mathcal{L}_2^0 - \mathcal{L}_2^*}{T} + \frac{L\gamma^2}{2} \left[ \frac{M_1}{1-\eta_2} + \frac{M_2(\eta_1 - \eta_2 + 1)}{1-\eta_2} \right]. \quad (4.26)$$

Clearly, we can wrap  $k = 1, 2, \dots, k-1$  to an integral constraint and apply above analysis. The wrapped constraints also follow the sub-linear bound, thus the result hold.  $\blacksquare$

### 4.5.2 Proof of Theorem 2

*Proof.* The estimator is unbiased,  $\mathbb{E}_{\hat{n}_u} \left[ \sum_{v \in \hat{n}_u} \hat{P}_{uv} \mathbf{x}_v \right] = \sum_{v \in n_u} P_{uv} \mathbf{x}_v$ , and let the sampling probability  $\{q_v\}$  and the corresponding normalization term  $c$ , then the expectation of the estimator can be written as,

$$\mathbb{E}_{\hat{n}_u} \left[ \sum_{v \in \hat{n}_u} \bar{P}_{uv} \mathbf{x}_v \right] = \mathbb{E}_{\mathbb{I}_v} \left[ \sum_{v \in n_u} c \mathbb{I}_v P_{u,v} \mathbf{x}_v \right], \quad (4.27)$$

where  $\mathbb{I}_v$  is the indicating random variable, with 1 indicating the occurrence of  $v$  in the sample  $\hat{n}_u$  and 0 otherwise.

Next, the variance of the estimator with regard to  $\{q_v\}$  is,

$$\text{Var}_{\hat{n}_u}(\hat{\mathbf{P}}_u \mathbf{x}_u) = \mathbb{E}_{\mathbb{I}_v} \left[ \left\| \sum_{v \in n_u} c \mathbb{I}_v P_{uv} \mathbf{x}_v - \sum_{v \in n_u} P_{uv} \mathbf{x}_v \right\|^2 \right] = \mathbb{E}_{\mathbb{I}_v} \left[ \sum_{v \in n_u} \|\dot{\mathbf{h}}_v\|^2 + \sum_{\{v_1, v_2\} \subseteq n_u} \dot{\mathbf{h}}_{v_1} \dot{\mathbf{h}}_{v_2} \right] \quad (4.28)$$

where  $\dot{\mathbf{h}}_v \doteq c \mathbb{I}_v P_{uv} \mathbf{x}_v - P_{uv} \mathbf{x}_v$ , and the first term follows

$$\mathbb{E}_{\mathbb{I}_v} \left[ \|\dot{\mathbf{h}}_v\|^2 \right] = \mathbb{E}_{\mathbb{I}_v} \left[ ((c \mathbb{I}_v - 1) P_{uv} \mathbf{x}_v)^2 \right] = \mathbb{E}_{\mathbb{I}_v} \left[ (c \mathbb{I}_v - 1)^2 \right] (P_{uv} \mathbf{x}_v)^2, \quad (4.29)$$

and the second term  $\mathbb{E} \left( \sum_{v_1, v_2 \in n_u} \dot{\mathbf{h}}_{v_1} \dot{\mathbf{h}}_{v_2} \right) = 0$  follows by Assumption 2. Thus, the variance of the estimator is given by,

$$\text{Var}_u(\bar{\mathbf{P}}_u \mathbf{x}_u) \leq \sum_{v \in n_u} \mathbb{E}_{\mathbb{I}_v} \left[ (c \mathbb{I}_v - 1)^2 \right] \sum_{v \in n_u} (P_{uv} \mathbf{x}_v)^2 = \left( \sum_{v \in n_u} \mathbb{E}_{\mathbb{I}_v} (c \mathbb{I}_v)^2 - \|V\| \right) \sum_{v \in n_u} (P_{uv} \mathbf{x}_v)^2, \quad (4.30)$$

where the last equality follows the definition of  $c$  and  $\mathbb{E}_v, \sum_{v \in V} c \mathbb{I}_v = \|V\|$ .  $\sum_{v \in n_u} \mathbb{E}_{\mathbb{I}_v} (c \mathbb{I}_v)^2$  is upper bounded by  $c \mathbb{I}_v > 0$ . ■

## 4.6 Experiments

In this section, we validate the effectiveness of the proposed method. We describe the experiment settings in 4.6.1, include the model accuracy and running time per epoch results in 4.6.2, present the convergence results in 4.6.3, and demonstrate the versatility of the proposed method in 4.6.4.

### 4.6.1 Experiment Settings

We evaluate the performance of the proposed method empirically on the following benchmarks tasks: Cora, Citeseer and Pubmed. For accuracy and running time comparison, we evaluate the performances on both the supervised tasks and semi-supervised tasks. In the supervised tasks, we use all labelled data as the training set. In the unsupervised tasks, we adopt a coherent split of data used in [64] for fair comparison. For other comparisons we only report the results on supervised tasks. A GCN model with one hidden layer is implemented for all datasets, and we report the results trained using different accelerating methods. We compare the proposed methods to related baselines, including stochastic methods without sampling (full), neighbor sampling (NS) [50], and importance sampling (IS) [24]. Based on naive GCN [64], batch normalization [60] are added to the model besides the last layer, to reduce the sample variance [23]. The statistics of the involved datasets, the corresponding model parameters and the training details are summarized in Table. 11. Here the rows *semi-supervised* and *supervised* are the sizes of training set for the two tasks, respectively.

The proposed method is implemented in Tensorflow [2]. In the proposed method,  $K$  is set to 2, and the neighbor sampling is used. We use cross-entropy loss and Adam optimizer [63] throughout the experiments, and the step-size is fixed to 0.01. All experiments are performed on a server with one Intel i-7 CPU and three Titan X (PASCAL) GPUs.

### 4.6.2 Comparison with Baselines

In this section we report the model accuracy and running time on different training methods. Table. 12 summarizes the running time and test accuracy on the aforementioned



supervised and semi-supervised tasks and the three datasets regarding different methods.

For the supervised tasks, the epoch running time of naive GCN grows extremely fast with regard to the graph size. Stochastic methods can make the process more efficient, through training on induced sub-graphs. NS and IS use sampled sub-graphs, which can slightly reduce the running time in our experiments. With denser and larger graphs, the time efficiency can be further improved. Meanwhile, large sample size will lead to longer training time. Notably, the proposed method significantly reduces the training time for all datasets. This indicates that the proposed method can provide solid improvements in faster training of GCN, based on existing methods. In fact, in the proposed method, multiple sub-models accomplish the sampling simultaneously, and with adequate CPU resources, the acceleration on large dataset, for example Pubmed can be further promoted.

For accuracy results, the model trained without stochastic methods are also included to establish the “expected” performance. The model trained using stochastic methods on induced sub-graphs yields slightly worse results than exact methods. The model trained using the proposed method attains almost identical results to the stochastic methods on most datasets. Moreover, the proposed method has smaller generalization error on some datasets. To sum up, the convergence of the proposed method is confirmed by these results, and the performances of both time efficiency and model accuracy are demonstrated.

For the semi-supervised tasks, similar results can be confirmed: the proposed method consumes much less training time, and attains at least comparable generalization ability compared to baselines. Notably, IS appears to be unstable in the semi-supervised tasks, due to the trivial zero activation issue. However, the proposed method using IS can achieve a much better generalization ability. Empirically, it implies that the proposed method can stabilize the training with IS.

### 4.6.3 Convergence Results

In this section we show the convergence results of the proposed method empirically. Figure. 14 gives the train loss tendency against epochs for supervised tasks. For Cora and Citeseer, the loss decreasing of NS and IS are similar; for Pubmed, IS attains a slightly

smaller loss compared to NS. From the results we can find that, for a given sample scheme, the proposed method has a identical convergence rate with naive stochastic methods. In the long run, the loss curves of naive stochastic methods with either NS or IS are oscillating occasionally during training, but the proposed method are more stable.

Figure. 13 depicts the accuracy trending on test sets against epochs. Again, with given sampling schemes, the proposed method can yield the same results with naive stochastic methods, but costs much less time. These results demonstrate that the proposed method has the same sub-linear convergence rate and generalization abilities with naive stochastic methods, as well as providing significant efficiency improvement.

#### 4.6.4 Efficiency Under Various Settings

The proposed method is independent of sampling method and network structure. In this section we show the efficiency of the proposed method with various settings.

Figure. 15 shows the running time per epoch with different depths and  $K$ . For 3-layer GCN, the hidden unit numbers are 32 and 16 respectively. When  $K = 2$ , we split the first two layers and the last layer as two sub-models; when  $K = 3$ , we split each layer as a sub-model. For 4-layer GCN, the numbers of hidden units are 32, 32, and 16 respectively. When  $K = 2$ , we split the first two layers and the last two layers as two sub-models; when  $K = 3$ , we split the first two layers, the third layer, and the last layer as three sub-models. We only include the results using NS, because IS has a similarly performance. With the network growing deeper, the running time per epoch increases rapidly. The proposed method can greatly reduce the training time to almost half of the naive stochastic methods, for different network depth. Different  $K$  corresponds to different number of sub-models, and larger  $K$  yields a similar results to smaller  $K$ . This is due to the bottleneck of CPU, which handles the sampling computation.

## 4.7 Conclusion

In this paper we proposed a fast training method for GCN. We split the naive GCN model into sequential sub-models, and re-formulated the optimization problem. With gradient flashback, multiple sub-models can sample sub-graphs and update model parameters simultaneously. We also provided a theoretical analysis on the convergence rate for non-convex problems. The experiments showed that the proposed method can be assembled with different model structures and sampling methods, and share consistent convergence rate and generalization ability with naive stochastic methods. Particularly, the proposed method consumes much less time per epoch thus accelerating the training, and can stabilize the training of stochastic methods.

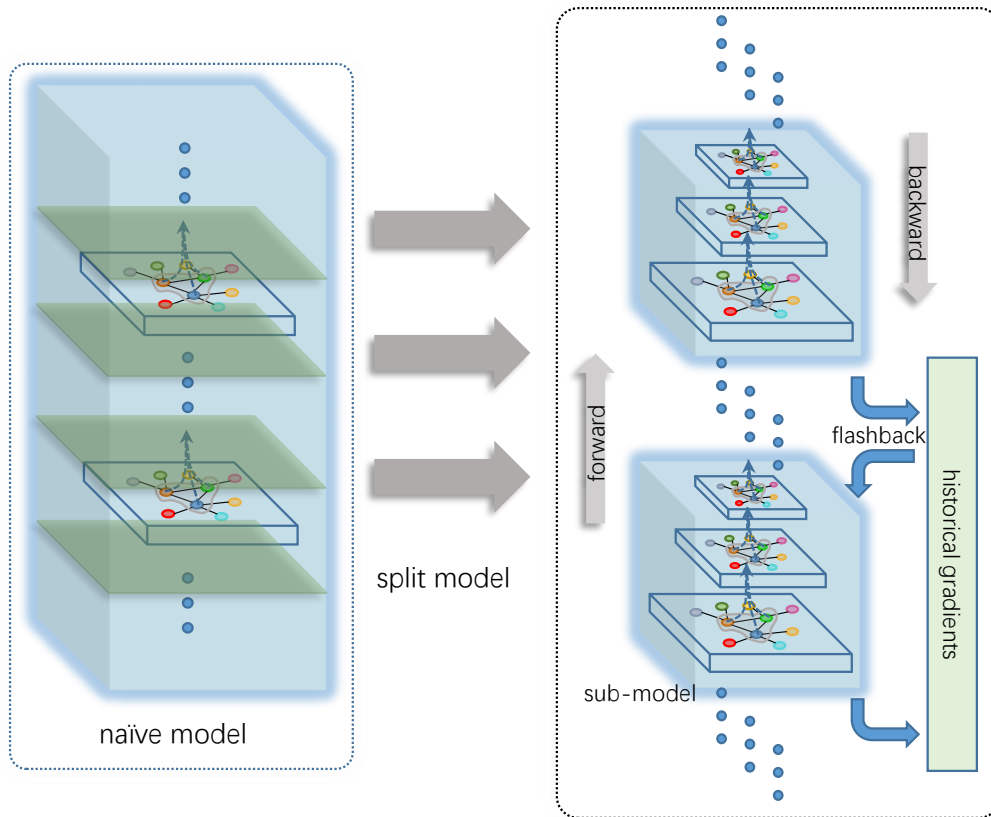


Figure 12: The training process of the proposed method. Naïve GCN model are splitted into multiple sequential sub-models. In the forward step, the outputs are computed using current model; in the backward step, the updating is computed using stale error gradients. We name it *gradient flashback* due to the re-occurrence of historical gradients.

Table 11: Dataset description and model/training details.

dataset	Cora	Citeseer	Pubmed
nodes	2708	3327	19717
edges	5429	4732	44338
classes	7	6	3
features	1433	3703	500
semi-supervised	140	120	60
supervised	1208	1812	18217
hidden units	16	16	16
batch size	128	128	128

Table 12: Comparison of different training methods on multiple tasks and multiple datasets. Top: running time of different methods, the best results are marked in bold font; bottom: test accuracy of the models trained using different methods. The time efficiency and consistent performance can be observed from the values.

	dataset	Supervised			Semi-Supervised		
		Cora	Citeseer	Pubmed	Cora	Citeseer	Pubmed
running time	full	0.5346	1.2111	6.7703	0.2045	0.3189	0.2122
	NS	0.4242	1.1356	2.6538	0.1866	0.2994	0.1914
	IS	0.4246	1.1423	2.6366	0.1743	0.2870	0.1896
	proposed	<b>0.1237</b>	<b>0.4665</b>	<b>1.5527</b>	<b>0.0283</b>	<b>0.0663</b>	<b>0.0297</b>
test accuracy	naive	0.8690	0.7720	0.8740	0.7810	0.6370	0.7670
	full	0.8560	0.7692	0.8869	0.7613	0.6225	0.7465
	NS	0.8457	0.7571	0.8667	0.7879	0.6479	0.7336
	IS	0.8590	0.7672	0.8432	0.6817	0.5852	0.6913
	proposed	0.8550	0.7690	0.8720	0.8050	0.7080	0.7410

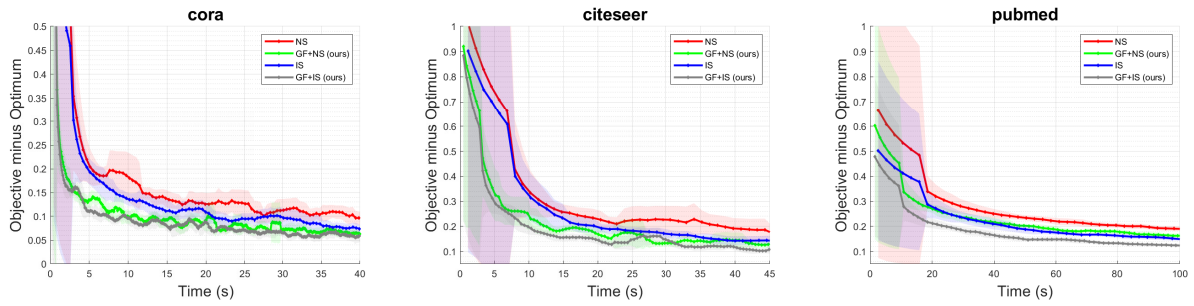


Figure 13: The results of loss against time on training set. For all datasets, the proposed method converges much faster than naive methods, given the same sampling scheme. In the long run, it also attains an identical performance with naive stochastic methods, which verifies the convergence analysis.

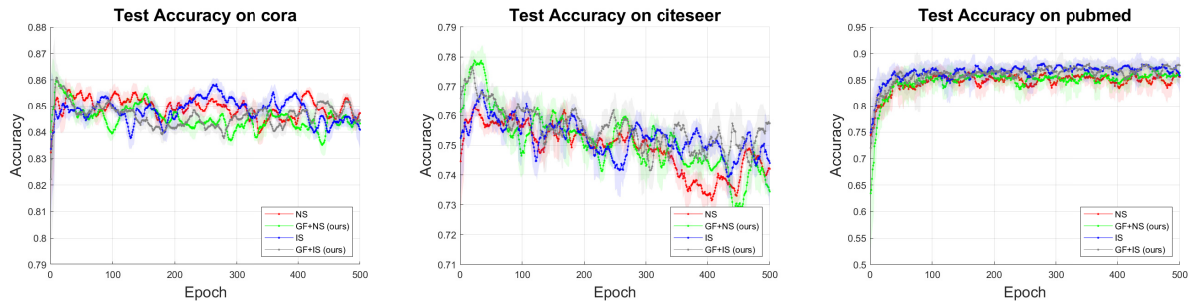


Figure 14: The results of accuracy against epoch on test set. For all datasets, the proposed method attains an identical performance with naive stochastic methods, given the same sampling scheme.

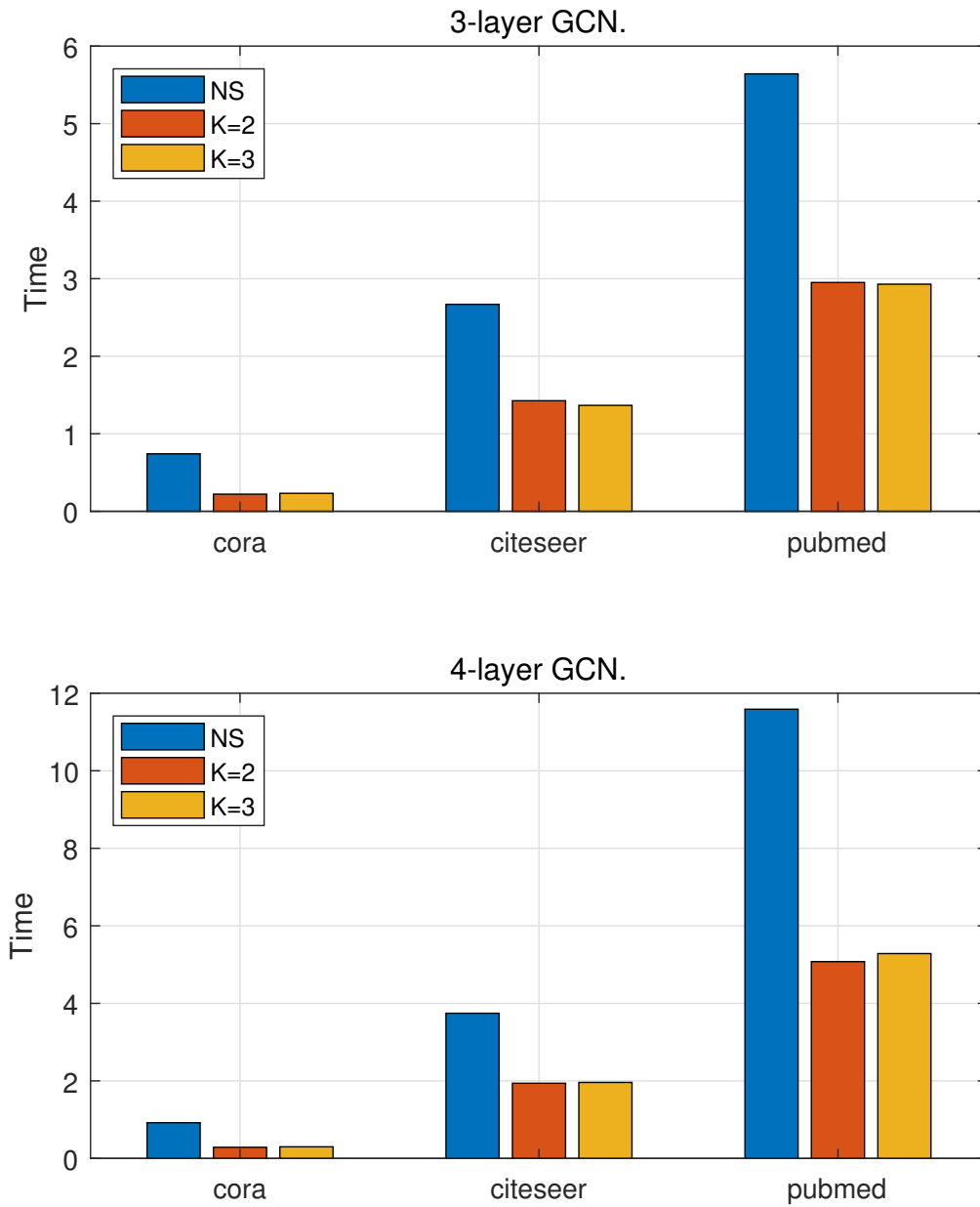


Figure 15: Running time per epoch. The proposed method consumes much less time compared to naive stochastic methods using the same sampling scheme.

## 5.0 Applying Graph Representation Learning to Varied Medical Imaging Problems

This section will discuss some graph-based representation methods designed for various applications [20, 170, 171, 43, 172, 139, 83, 182, 179, 174, 147, 178, 176, 140, 46, 175, 169].

### 5.1 Learning Shared Structure from Single-view Graph data

#### 5.1.1 Motivation

Connectome describes the neural connection within a brain [117]. Utilizing state-of-the-art neuroimaging techniques including functional resting state magnetic resonance imaging (rs-fMRI) [28] and structural diffusion tensor imaging (DTI) [71], it is possible to capture the connectome at macroscopic scale, which refers to the scenario involving anatomically segregated brain regions and inter-regional pathways. By graph based analysis, the information encoded by the connectome can promote critical understanding on how the brain manages cognition, what signals the connections convey and how these signals affect brain regions [42]. It has been discovered to be helpful in the early diagnosis of several neurological disorders, including epilepsy, Alzheimer's disease, and autism [134, 80, 44, 150]. Across depressed patients and normal subjects, functional brain connectomes derived from rs-fMRI also display distinct patterns [155, 129, 168], and some computational methods have been proposed to study the relation, for example linear support vector machine (SVM) [160], partial least square (PLS) [155]. Deep learning methods, which are successful on many tasks, are exploited as well. In [138], convolution neural network for classification is applied to brain networks with nodes reordered using a spectral clustering method.

Although meaningful clinical representations can be obtained from brain networks through previous methods, some issues still exist. The human connectome has sophisticated and non-linear structure, which may not be well captured by shallow linear models. Meanwhile, deep



learning methods suffer from the enormous parameter sizes, which is both difficult for training and vulnerable to over-fitting potentially. Besides, many methods do not make good use or even fail to preserve the graph structure. Thus, a concise graph based deep learning method is preferred on the task to discover the relation between human connectome and clinical scores. In the studies on graph data, great efforts are spent on node embeddings, with the majority of which are based on graph spectral properties and random walk techniques [49, 102, 12, 133]. Recently, several convolutional neural network methods for graph data are proposed [7]. Particularly, the graph convolution network (GCN) [30] based on spectral graph theory and its variants [64] are flexible in both graph and node analysis. However, GCN explicitly requires a known graph structure, which is typically not available in brain connectome. To address this challenge, in this paper we studied the GCN with *unknown* graph structure. In detail, we showed that GCN without given graph structure is applicable, by using a naive complete graph. Meanwhile, a method was propose to learn the graph structure from data to improve the performance of GCN, by generating random graph with small-world property for model training. The experiments demonstrates that the proposed method outperforms related baselines in the prediction of clinical depression scores, and the learned graph is superior to the naive complete graph settings.

The rest of this paper is organized as follow. §5.1.2 provides the preliminary and describes the detail of the proposed method. §5.1.4 shows the experiments and the results. §5.1.5 concludes the paper.

### 5.1.2 Methodology

To address the problem of predicting clinical depression scores, we proposed a two-stage regression method. The first stage is to obtain the embeddings of a single connectome, involving a graph convolution network that can be applied to data without predefined precise graph structure; the second stages is merely a standard MLP, for regression based on the embeddings. We introduced the the preliminary on GCN in §5.1.2.1, and considered the GCN formulation without predefined graph structure in §5.1.2.2. In §5.1.3 the entire algorithm is given and some details are discussed.

### 5.1.2.1 Preliminary

A graph can be represented as  $\{V, E, W\}$ , with  $V = \{v_1, v_2, \dots, v_n\}$  the set of  $n$  vertices,  $E \subseteq V \times V$  the set of  $m$  edges, and  $W$  the weighted adjacency matrix of the graph. The adjacency matrix  $W \in \mathcal{R}^{n \times n}$ , and  $W_{ij}$  encodes the relation between  $v_i$  and  $v_j$  if there is an edge, and 0 otherwise. The degree matrix of a graph can be expressed with diagonal matrix  $D \in \mathcal{R}^{n \times n}$ , with the diagonal entries  $d_{ii} = \sum_j w_{i,j}$ , representing the degree of node  $i$ . Graph Laplacian is an essential operation in spectral graph analysis, typically defined as  $L_c = D - W$  in combinatorial form and  $L_n = I_n - D^{-1/2}WD^{-1/2}$  in normalized form, with  $I_n$  a identity matrix. Graph convolutional network (GCN) [30] is designed to analysis the signals on nodes, with a given graph structure. As the name indicates, GCN is an extension of convolutional neural network (CNN). Convolution on 2D matrices, for example images, is well defined, serving as the basic building block in CNN. Graph convolution, however, is difficult to perform directly in spatial domains, due to the irregularity of neighbors of distinct nodes. One strategy is to conduct the operation in frequency domain using graph Laplacian, which is feasible according to the convolution theorem [84]. By definition, Laplacian matrix is positive semi definite, such that the eigenvalue decomposition exists,  $L = U^T \Lambda U$ , and  $U = [u_0, u_1, \dots, u_{n-1}]$  specifies a Fourier basis. The graph Fourier transform [115] is then defined as  $\hat{x} = U^T x$ , with  $x \in \mathcal{R}^n$  the signal, and the inverse transform  $x = U \hat{x}$ . The spectral representation of node signals,  $\hat{x}$ , allows the fundamental filtering operation for graphs.

One potential problem in the above formulations is that in spectral domain, the filter is not naturally localized. Polynomial parametrization for localized filters [30] is proposed to tackle this challenge, through learning the coefficients  $\Theta_K$  of a  $K$ -order Chebyshev polynomial. In detail, the filter is defined as  $g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\Lambda)$ , and the graph convolution is defined as  $y = U g_\theta(\Lambda) U^T x$ , with  $y$  the filtered signal,  $\theta_k$  the trainable parameters and  $T_k(\Lambda)$  the polynomials. By the stable recurrence relation of Chebyshev polynomial, the costly multiplication in Fourier transform can also be reduced. Parallel to CNN, pooling operation in graph settings is accomplished by the graph coarsening [33] procedure.

The above method is suitable for graph classification, which is extended to node classification later [64]. By truncating the Chebyshev polynomial to only first order, GCN

model can be reformulated similar to multi-layer perceptron, with each layer defined as  $y = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}x\Theta)$ , with  $\tilde{A} = A + I_n$ ,  $\tilde{D}$  accordingly defined as  $D$ ,  $\Theta$  the trainable parameters, and  $\sigma(\cdot)$  an activation function. In the proposed method we will adopt the first order approximation version of GCN as the first stage to obtain embeddings for a graph, then use a MLP in the second stage for the regression task.

### 5.1.2.2 Graph Convolutional Network without Pre-defined Graph Structure

Standard GCN, as well as its variants, defines the graph convolution based on a known adjacency matrix  $A$ . However, in our task we only know human connectomes are equipped with graph structure, while the precise graph is never provided. In this section we attempt to answer three questions: (1). can we apply GCN to graph data without predefined graph structure? (2). how can we find a “good” graph structure based on the task and the data? and (3). is GCN preferred compared to naive neural network?

### 5.1.2.3 Can we apply GCN to graph data without predefined graph structure?

The answer to the first question establishes the foundation of the solvability of the task. we will give the problem formulation and show that GCN is applicable under such formulation.

For a standard perceptron layer,

$$vec(y) = \sigma(vec(x)W) \simeq \sigma(vec(x)(\tilde{L}_g \otimes \tilde{\Theta})^T) = \tilde{L}_g x \tilde{\Theta}, \quad (5.1)$$

where  $vec(\cdot)$  is vectorization operator,  $x \in \mathcal{R}^{n \times n}$ ,  $W \in \mathcal{R}^{n \times c}$  with  $c$  the channels of output  $y$ ,  $\otimes$  Kronecker product, and  $\tilde{L}_g$  and  $\tilde{\Theta}$  the decomposition of  $W$ . If the eigenvalue decomposition of  $\tilde{L}_g$  exists,

$$\tilde{L}_g = U_g^T \tilde{\Lambda}_g U_g, \quad (5.2)$$

then  $y = \sigma(U_w^T \Lambda_w U_w x)$  is identical with GCN [64] by their form, thus

$$\tilde{\Lambda}_g = \theta_0 + \theta_1 \left( \frac{2}{\lambda_{max}} \Lambda_g - I_n \right), \quad (5.3)$$

where  $\Lambda_g$  is the eigenvalues of the underlying graph Laplacian  $L_g$ , and the graph Laplacian shares the same eigenvectors as  $\tilde{L}_g$ , thus  $L_g = U_g^T \Lambda_g U_g$ . Eq. (5.3) implies that given a

perceptron, an equivalent GCN layer can be induced. Essentially if  $L_g$  is given, we are inserting priors on the weight matrix  $W$  in perceptron, and the parameters of the model can be greatly reduced to merely  $\theta_0$  and  $\theta_1$ .

Following the above argument, we can formulate  $L_g$  by selecting meaningful priors if it is not given. Here we show a naive choice, which is effective in denoising sense. The input connectome is  $x = [x_1, x_2, \dots, x_n]$ , and We treat the column  $x_i \in \mathcal{R}^n$  as the signals for the corresponding node  $i$ . Assume the collected connectome data are noisy,

$$x = x_0 + \sigma_0, \quad (5.4)$$

where  $\sigma_0$  is a random noise matrix with entry-independent Gaussian distribution, and  $x_0$  is the clean graph. For single layer GCN,

$$y = (\theta_0 - \theta_1)x - \frac{2\theta_1}{\lambda_{max}}L_g x, \quad (5.5)$$

the first term is 0 is we choose  $\theta_0 - \theta_1 = 0$ , making no contribution to the output. For the second term, we want to select the  $L_g$  that minimizes the effect of noise  $\sigma_0$ ,

$$L_g = \arg \min_{L_g} E_{\sigma_0} \left( \left\| \frac{2\theta_1}{\lambda_{max}} L_g \sigma_0 \right\|_F^2 \right), \quad (5.6)$$

where  $\|\cdot\|_F$  is the Frobenius norm and  $E_{\sigma_0}(\cdot)$  is the expectation over  $\sigma_0$ . Let  $\frac{2\theta_1}{\lambda_{max}} \simeq 1$ . The decomposition  $L_g = U_g^T \Lambda_g U_g$  still exists.  $U_g$  is an orthogonal matrix and  $\sigma_0$  is Gaussian, thus  $U_g \sigma_0$  is still Gaussian. Let  $U_g \sigma_0 = [\sigma_1, \sigma_2, \dots, \sigma_n]$ ,

$$\begin{aligned} L_g &= \arg \min_{L_g} E_{\sigma_0} \left( \text{tr} \left( \sum_i \lambda_i^2 \sigma_i^T \sigma_i \right) \right) \\ &= \arg \min_{L_g} \sum_i \lambda_i^2 E_{\sigma_i} (n \times \text{tr} (\sigma_i^T \sigma_i)) \\ &= \arg \min_{L_g} \|L_g\|_F^2 n E_{\sigma_i} (\text{tr} (\sigma_i^T \sigma_i)) = \arg \min_{L_g} \|L_g\|_F^2, \end{aligned} \quad (5.7)$$

To this end we just need to choose the  $L_g$  with minimum Frobenius norm,

$$\begin{aligned} \|L_g\|_F^2 &= \text{tr}((I_n - D^{-1/2} A D^{-1/2})^2) \\ &= \text{tr}(I_n - 2D^{-1/2} A D^{-1/2} + D^{-1/2} A D^{-1} A D^{-1/2}) \\ &\geq \text{tr}(I_n) = n, \end{aligned} \quad (5.8)$$

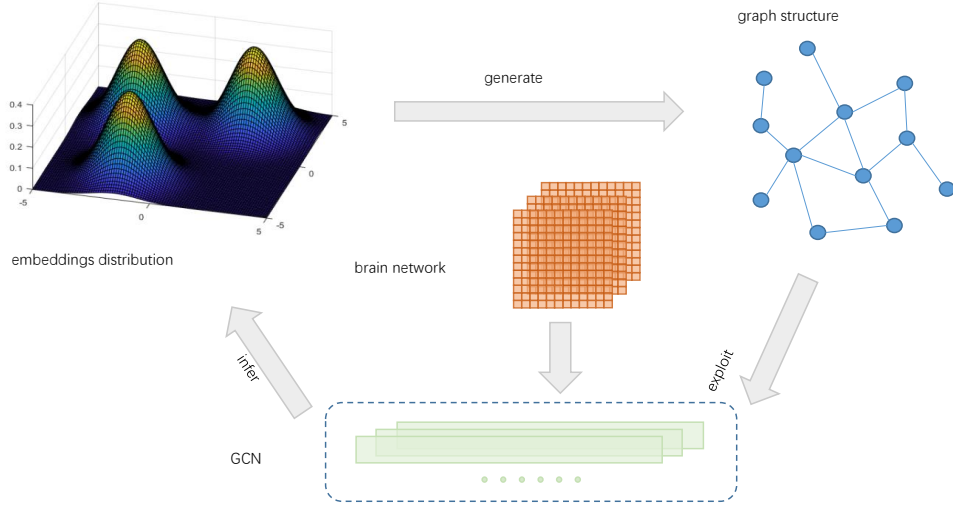


Figure 16: The iterative procedure of random graph generation, GCN training, and embeddings inference.

because  $\text{tr}(D^{-1/2}AD^{-1/2}) = 0$  and  $\text{tr}(D^{-1/2}AD^{-1}AD^{-1/2}) \geq 0$ . Let  $L_g^* = I_n - \frac{\mathbf{1}\mathbf{1}^T}{n-1}$ , then the minimum is attained.  $L_g^*$  provides the best denoising effect, given Gaussian noise. Not surprisingly, such a  $L_g$  corresponds to a complete graph, which is a naive choice of graph structure. Intuitively, it indicates that if nothing is known about the potential node relations, we can just assume all node are related equivalently.

Moreover, the above result also minimizes the kernel norm,

$$\arg \min_{L_g} E_{\sigma_0} \left( \left\| \frac{2\theta_1}{\lambda_{max}} L_g \sigma_0 \right\|_* \right) \equiv \arg \min_{L_g} E_{\sigma_0} \left( \left\| \frac{2\theta_1}{\lambda_{max}} L_g \sigma_0 \right\|_F \right), \quad (5.9)$$

which further strengthens the advantages of the naive complete graph choice.

#### 5.1.2.4 How can we find a “good” graph structure for brain connectome?

Although the naive as well as intuitive choice of complete graph is sufficient for alleviating Gaussian noise, the real-life data are sophisticated and more flexible methods are expected. This boils down to the importance of the second problem.

When different graph Laplacian is used, the obtained GCN will has different parameters and the outputs are generally different. Assume two GCNs are trained on the same data using  $L_1$  and  $L_2$  respectively, and for one layer the parameters are  $\Theta_1$  and  $\Theta_2$  accordingly, to make the two GCNs to yield the same outputs, we need,

$$L_1 X \Theta_1 = L_2 X \Theta_2, \quad (5.10)$$

which can be re-formulated as,

$$L_2^{-1} L_1 X - X \Theta_2 \Theta_1^{-1} = \mathbf{0}, \quad (5.11)$$

the equation is a Sylvester function with fixed  $L_2^{-1} L_1$  and  $\Theta_2 \Theta_1^{-1}$ , and the solution set  $X$  is the data. To fit the data  $L_2^{-1} L_1$  and  $\Theta_2 \Theta_1^{-1}$  must have common eigenvalues, which is possible only when the network is wide, namely  $\Theta_1, \Theta_2 \in \mathcal{R}^{d \times c}$ , and  $c \geq d$ . This usually will not happen in realistic models. Therefore, selecting graph structure is necessary.

A “good” graph structure is supposed to have two properties: ideally it should reflect the structure of data; also, it should be stable, in the sense that networks with similar graph structures should share similar parameters and outputs. Intuitively, we can initialize the graph Laplacian with high correlations in brain connectomes, and update the network in order to exploit the data structure. However, the naive back-propagation method in training neural networks is not applicable, because unlike normal neural network parameters, graph Laplacian is a structural matrix, satisfying some particular properties. Instead of solving the complex constraint optimization problem, we propose a random generated graph Laplacian using small-world model during the training, to infer the underlying data structure.

In the proposed method GCN is used to extract the embeddings of connectomes, on which a popular assumption is that its prior distribution is known. Further, we assume distributions are associated with the graph structure. For simplicity, we assume the underlying model is known, and our target is to obtain embeddings characterizing both the dependent variables and the graph structure. Therefore we can apply a loop structure during training, which is illustrated in Fig. 16: novel random graph is iterative generated, used as the graph structure training the current model; meanwhile, the latent embeddings are optimized with the evolving of the random graph. The training ultimately yields stable embeddings, and we use the

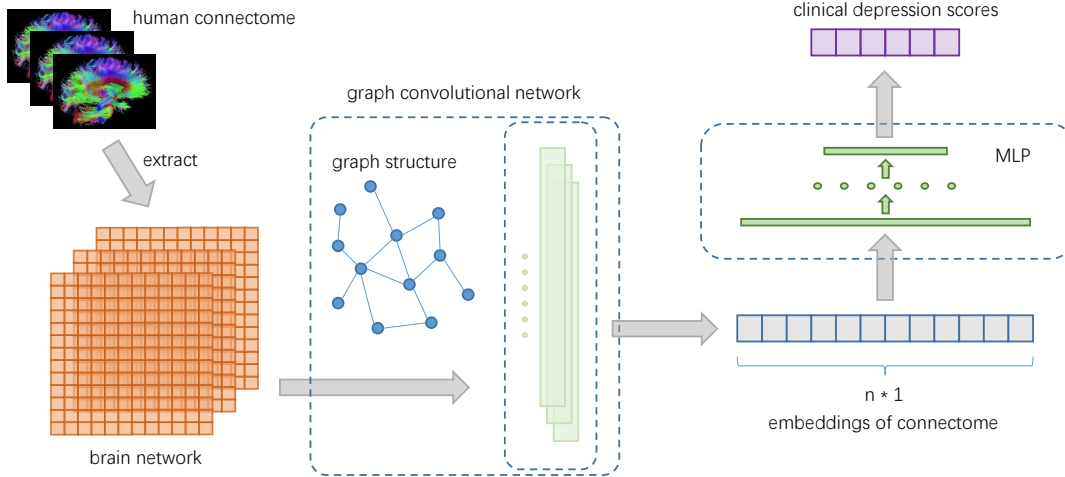


Figure 17: The structure of the proposed method. Brain connectome is transformed into embeddings using GCN and MLP is used for regression successively.

expectation of the generated random graph in the inference, which can be viewed as a “good” graph structure, because it integrates the average results of the embeddings.

In this paper, we use the small-world network to relate the latent embeddings to the graph structure. In this model, the distance between most nodes are a small number of hops or steps, and nodes with common neighbors are more likely to be directly connected. Meanwhile, human brains are also found to display some properties of this model [18, 10], which provides an intuitive interpretation: ROIs commutes information through small-world like networks, whose behaviors are simulated by the GCN. The embeddings of connectome are  $h \in \mathcal{R}^n$ , and the probability of the existence of an edge between node  $i$  and  $j$  is [75],

$$p_{e_{ij}} = \frac{\epsilon_p}{|h_i - h_j|^{\delta_p} + \epsilon_p}, \quad (5.12)$$

here  $\delta_p$  and  $\epsilon_p$  are hyper parameters. During training random graph structure is generated using Eq. (5.12), meanwhile the graph parameters are adjusted according to the embeddings. We apply K-means algorithm to cluster the clinical depression scores and obtain the centers of the embeddings for each cluster during the loop. The random graph is then generated

by sampling these centers. The cluster of each subject is called “pseudo-label” and kept unchanged throughout.

### 5.1.2.5 Is GCN preferred compared to naive neural network?

Exploiting neural network in node classification has been actively studied. The extension to multi-task regression problem on graph data thus is natural. Naive neural network methods flatten the graph data into vectors and build a multi-layer perceptron for the regression task, which involves enormous amounts of parameters: the dimension of a single input is  $n^2$ , and the hidden units are decided accordingly. In the proposed model, the parameter sizes are significantly reduced. Meanwhile, the number of hidden units for both stages can be drastically decreased. Though over-parameterization has some potential benefits [35], it makes the model difficult for training, particularly in our task with limit data.

Sparsity is highly effective in solving high-dimension machine learning tasks. Unfortunately utilizing sparsity structure is not trivial in neural network models. The first-stage of the proposed method, in which the graph data is reduced to vector embeddings, can be viewed as dimension reductions. Particularly, unlike “black-box” encoder models with elusive representations, the connectome embeddings are explicit connected to the graph structure of human brains, and clustered according to the depression scores, which is more explainable.

### 5.1.3 Pseudo-Label

To this end we achieve approval answers on the three questions and in this section the entire method is given. The proposed method is composed of two stages. In the first stage a GCN is used to extract the embeddings of a connectome, and in the second stage a MLP is used for the regression using the embeddings. The model’s structure is illustrated in Fig. 17.

The proposed method is straight-forward if the naive complete graph is used in the GCN. If the graph structure is to be learned, the proposed method explicitly requires data “labels” for inferring the graph structure. In this paper we use K-Means to generate the pseudo data “labels” based on the dependent variables during training. The objective is,



---

**Algorithm 4** Training GCN without pre-defined graph structure

---

**Input:** training set  $x$ , validation set  $x_v$ , clinical depression scores  $y$ , hyper parameters  $\delta_p, \sigma_p$ .

**Output:** the embedding of the graph and the trained model

```
1 Initialization;
2 for  $k$  do
3   K-Means, generate pseudo labels  $c$ , w.r.t  $y$ ;
4   repeat
5     while epoch not end do
6       Obtain embeddings  $h$ , i.e. the output of GCN;
7       Estimate center for each cluster;
8       Generate a random graph w.r.t. Eq. (5.12);
9       Train GCN and MLP w.r.t. Eq. (5.19);
10    end
11    Generate expected graph laplacian;
12  until converge;
13 end
14 Choose best model on validation set.
```

---

$$\mathcal{L} = \|\hat{Y} - Y\|_F + \lambda_1 \|W\|_2 + \lambda_2 \|\hat{H} - H\|, \quad (5.13)$$

where  $\hat{Y}$  and  $Y$  are estimated and ground truth clinical depression scores respectively,  $W$  is the model parameter,  $\hat{H}$  and  $H$  are estimated and ground truth connectome pseudo-labels respectively, and  $\lambda_1$  and  $\lambda_2$  is a tunable parameter. The optimization generally follows the procedure of EM algorithm, because the embeddings are sufficient statistics for inferring the graph structure. The algorithm involving learning the graph structure is summarized in Algorithm. 4.

K-Means method is known to be sensitive to parameters and converges only to local minimum. In our experiments we found that with fixed  $k$  and random initialization occasionally

the graph structure converge to unsatisfactory results. To achieve the best performance, we can use multiple starts with different  $k$  in the K-Means step and choose the best model with regard to validation data.

## 5.1.4 Experiments

### 5.1.4.1 Data Description

We conducted the experiments to predict clinical depression scores using the Human Connectome Project (HCP) data ([www.humanconnectomeproject.org](http://www.humanconnectomeproject.org)). The fMRI measurements were obtained on a 3T GE Signa HDx scanner with a 2D EP/GR. The subjects were asked to not take psychotropic drugs before experiments. The images were acquired when they laid down with eyes open, kept awake and thought of nothing, after-while processed following SPM8 standard procedures. Different brain regions with different resolutions were defined based on voxels, according to the automatic anatomical labeling atlas [126]. Between each pair of regions, functional connectivity were computed using the cross correlation of the corresponding time-series.

The rs-fMRI signals involving 1000 subjects are obtained with different ROI resolutions. Through our experiment, we use the connectome with resolution of 50 and 100 node graphs. For all subjects, eight measures of clinical depression scores are used as the dependent variables, including *ASR Anxious/Depressed*, *ASR Thought Problems*, *ASR Attention Problems*, *ASR Aggressive Behavior*, *ASR Rule Breaking Behavior*, *ASR DSM Depressive Problems*, *ASR DSM Anxiety Problems* and *ASR DSM Antisocial Personality Problems Raw Scores*. In the pre-processing, we normalized the depression scores to zero mean and unit variance, and randomly split subjects to the training, validation, and test sets, respectively using 70%, 10%, and 20% of the data. We repeated the allocation 5 times and reported the mean and standard error of mean absolute error (MAE).

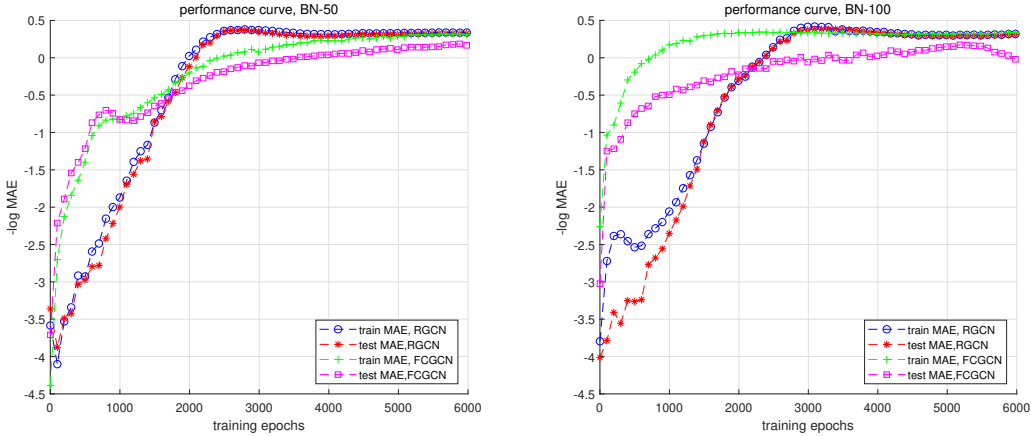


Figure 18: Comparison of training curves of FCGCN and RGCN.

#### 5.1.4.2 Experiment Setting

We compare the proposed method with several other regression methods: multivariate Ridge Regression (RR), Least Absolute Shrinkage and Selection Operator (LASSO), Elastic Net (EN), which is the combination of  $L_1$  and  $L_2$  norm, and Multi-Layer Perceptron (MLP). For RR, LASSO, and EN, the hyper-parameter was tuned on validation data. For MLP, we used a two layer structure and kept the hidden layers of the size with the embeddings of the proposed method. Through the paper, the proposed method is adapted from the fast GCN using first-order approximation. In this section, we provide both the results using Fully-Connected Graph Convolution Network (FCGCN), and Random generated Graph Convolution Network (RGCN). Besides, we also provide the results of simple sparse graph (kGCN), keeping top 50% strongest edges of the average correlations. In RGGCN, we use a two layer GCN in the first stage, with 10 hidden units, and also a two layer MLP in the second stage, with 20 hidden units. Leaky ReLU is exploited as the activation function. For the random graph, we use  $\delta_p = 2$  and  $\sigma_P = 0.01$ , and run six independent K-Means with  $k$  ranging from 3 to 8, respectively. The results are reported based on the validation set. The RMSProp optimization algorithm is used with learning rate of 0.001. For FCGCN and kGCN, identical parameter settings are utilized, except that the graph structure is fixed as a

Table 13: Quantitative comparison of baselines and the proposed method. Both the mean and the standard error are given, and the best results are bold faced. Metrics without significant difference between baselines and FCGCN are denoted with  $\star$ ; the metric without significant difference between FCGCN and kGCN/RGCN is denoted with  $\diamond$ .

Methods	MAE, BN-50	MAE, BN-100
RL	$1.1176 \pm 0.1723$	$1.2510 \pm 0.1625$
LASSO	$0.9996 \pm 0.1783$	$1.0078 \pm 0.1025$
EN	$0.9538 \pm 0.1616$	$0.9476 \pm 0.0773$
MLP	$0.8202 \pm 0.1258^\star$	$0.7690 \pm 0.0339^\star$
FCGCN	$0.8176 \pm 0.1656$	$0.7704 \pm 0.0220$
kGCN	$0.8397 \pm 0.0864^\diamond$	$0.7449 \pm 0.0516^\diamond$
RGCN	<b><math>0.7372 \pm 0.0684</math></b>	<b><math>0.7226 \pm 0.0579^\diamond</math></b>

complete graph or a simple sparse graph.

### 5.1.4.3 Performance Comparison

The results of the proposed method using both complete graph and learned graph are listed in Table. 13. From the table we can find that the proposed method constantly outperform linear baselines on all scales. This results imply that non-linear models are potentially more suitable for our task. Particularly, FCGCN yields comparable results with MLP, using much less parameters, which demonstrates the advantages of GCN models. kGCN shows improvements against FCGCN w.r.t. MAE, but worsens MSE. Meanwhile, the proposed method using trainable graph structure has better performance than complete graph case, showing the importance of graph structure selection and the effectiveness of the proposed algorithm.

We also want to mention that with the random generated graph structure, the training of the proposed method actually is faster than the fixed scenario. The training curve is

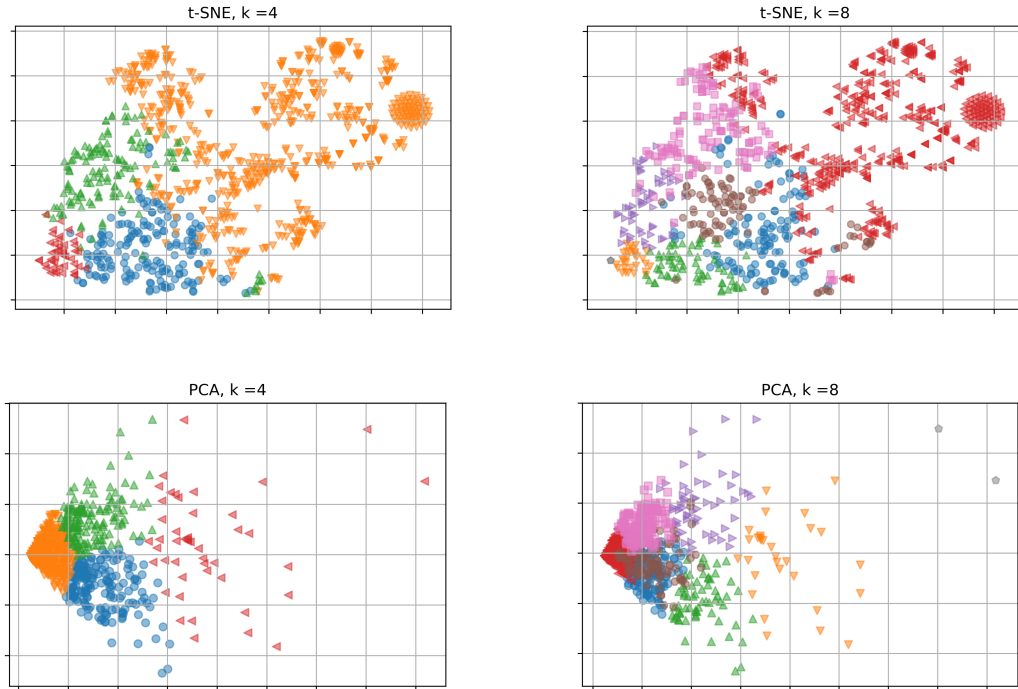


Figure 19: Visualization of clustering of clinical depression scores using t-SNE and PCA respectively, from a random start. The clustering is obtained through K-Means with  $K = 4$  and 8

illustrated in Fig. 18, in which RGCN provides a higher prediction ability and a smaller generalization error. This results further demonstrate that the learned structure fits the data better thus yields better performances.

In this paper we use K-Means as the clustering algorithm, which, though serves solid functions in our experiments, may not be stable. To depict this phenomenon, we present the clustering results by dimension reductions using both t-SNE and principle component analysis, and the results are listed in Fig. 19. Apparently, the clustering may not be intuitively preferable. That explains the necessity to improve the robustness of the proposed method through multiple start K-Means.

### 5.1.5 Conclusion

In this paper we proposed a novel method to predict the clinical scores using brain connectomes. We demonstrate that GCN can be applied to graph data even without pre-defined graph structure, and proposed an effective algorithm to learn the graph structure from the data. The experiments show the proposed method outperform standard methods, and particularly the learned graph structure yields a better results than the naive complete graph assumption.

## 5.2 Heterogeneous Graph Neural Networks Integrating Multi-view graphs

### 5.2.1 Motivation

Large-scale connection in the brains convey important insights in understanding the underlying yet unknown mechanism of many mental disorders [117, 18, 42]. With whole brain tractography, brain’s anatomical networks represented as major fiber bundles can be reconstructed from diffusion-weighted MRI (DWI). There are various brain networks, generated from different tractography algorithms based on either voxel-wise diffusion model or cross-voxel fiber tracking, each finding the place in revealing targeted brain abnormalities, such as autism spectrum disorder [68], Parkinson’s disease [22], and even in genetics. Nevertheless, for distinctive diagnosis tasks it is elusive to decide a universally optimal method and accompanied processing, e.g. dimension reduction [161], as that these tractography algorithms differ in the selection and accuracy of fiber extraction, robustness, and particularly the relevance between the extracted fiber bundles and the tasks. Essentially, tentative studies have demonstrated that multi-modal brain networks can provide complementary viewpoint toward the classification tasks, in leveraging the scattered information from acquisitions with diverse tractography algorithms. For example, it is showed that multi-view graph convolutional network [167] has state-of-the-art performance in classifying Parkinson’s disease (PD) status.

To take one step further, we propose to predict the clinical measures, instead of directly

classifying the disease status. The behind motivation lies in that, many mental disorders are degenerative, which can be inferred from the gradual progress of brain connectivity patterns, and that clinical measures, compared to simply classification, better capture the progress. Through integrating multi-modal brain networks in our prediction, a comprehensive assessment is constructed, and the potential deterioration from sub-optimal of single tractography is alleviated. To address the prediction problem, we resort to a cascade model, composed of a heterogeneous graph convolutional network (GCN) for brain network embeddings and a multi-layer perceptron (MLP) for regression. Our contributions are two-folded: first, we propose a heterogeneous GCN to predict the clinical scores from multi-modal brain networks, which benefits from the natural graph structures of diverse brain networks; second, an adaptive pooling scheme, driven by both graph structure and network patterns, is proposed, which is beneficial from gathering local information, yielding a faithful graph with smaller size, and enjoying efficiency in both computation and training. We name the proposed method as “heterogeneous” in that the graph convolution and pooling are customized for varied modal. The proposed method is verified on the data from the Parkinson Progression Marker Initiative (PPMI) [85], a cohort study aiming at identifying and validating PD progression markers. The experimental results show that our method outperforms related baselines significantly. It is also demonstrated that by integrating multi-modal brain networks, the proposed method achieves higher accuracy, and yields more stable prediction.

The rest of this paper is organized as follow. §5.2.2 provides the preliminary and describes the detail of the proposed method. §5.2.3 shows the experiments and the results. §5.2.4 concludes the paper.

## 5.2.2 Methodology

### 5.2.2.1 Preliminary

A graph can be represented as  $\{V, E, W\}$ , with  $V = \{v_1, v_2, \dots, v_n\}$  the set of  $n$  vertices,  $E \subseteq V \times V$  the set of  $m$  edges, and  $W \in \mathcal{R}^{n \times n}$  the weighted adjacency matrix of the graph. In this paper, vertices are Region Of Interest (ROIs). Graph Laplacian is an operation in spectral graph analysis [84], typically defined as  $L_c = D - W$  in combinatorial form and

$L_n = I_n - D^{-1/2}WD^{-1/2}$  in normalized form, with  $D \in \mathcal{R}^{n \times n}$  the diagonal matrix and  $I_n$  a identity matrix. Graph convolutional network (GCN) [30] is designed as an extension of convolutional neural network (CNN), to analysis the signals on nodes, with a given graph structure. One strategy is to conduct Graph convolution in frequency domain using the eigenvalue decomposition of graph Laplacian,  $L = U^T \Lambda U$ , and  $U = [u_0, u_1, \dots, u_{n-1}]$  specifies a Fourier basis. The graph Fourier transform [115] is then defined as  $\hat{x} = U^T x$ , with  $x \in \mathcal{R}^n$  the signal, and the inverse transform  $x = U \hat{x}$ . The spectral representation of node signals,  $\hat{x}$ , allows the fundamental filtering operation for graphs. For computational accessibility, polynomial parametrization for localized filters [30] is proposed through learning the coefficients  $\Theta_K$  of a  $K$ -order Chebyshev polynomial. The filter is defined as  $g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\Lambda)$ , and the graph convolution is defined as  $y = U g_\theta(\Lambda) U^T x$ , with  $y$  the filtered signal,  $\theta_k$  the trainable parameters and  $T_k(\Lambda)$  the polynomials. Parallel to CNN, pooling operation in graph settings is accomplished by the graph coarsening [33] procedure. By truncating the Chebyshev polynomial to only first order, a faster version of GCN can be reformulated [64] similar to multi-layer perceptron, with each layer defined as  $y = \sigma(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} x \Theta)$ , with  $\tilde{A} = A + I_n$ ,  $\tilde{D}$  accordingly defined as  $D$ ,  $\Theta$  the trainable parameters, and  $\sigma(\cdot)$  an activation function.

In graph classification, a major concern is to represent graphs with embeddings. Previous approaches [154] include averaging all the node embeddings in a final layer, computing “virtual node” connected to all nodes, operating over sets using deep node aggregation, concatenating all embeddings, and training hierarchical structure. A majority of these methods apply a deterministic graph clustering subroutine, while some end-to-end methods require additional structure to compute the pooling structure.

### 5.2.2.2 Predicting PD Clinical Scores via Heterogeneous GCN

The proposed method has two stages, as illustrated in Figure. 20. In the first stage, the per-modal embeddings for brain networks are generated via heterogeneous GCN; in the second, the concatenated embeddings are regressed to the clinical scores via MLP. Parallel to convolutional neural networks, the proposed GCN is formed by stacking graph convolutional layer and pooling layer sequentially. The fast graph convolutional [64] is applied, using



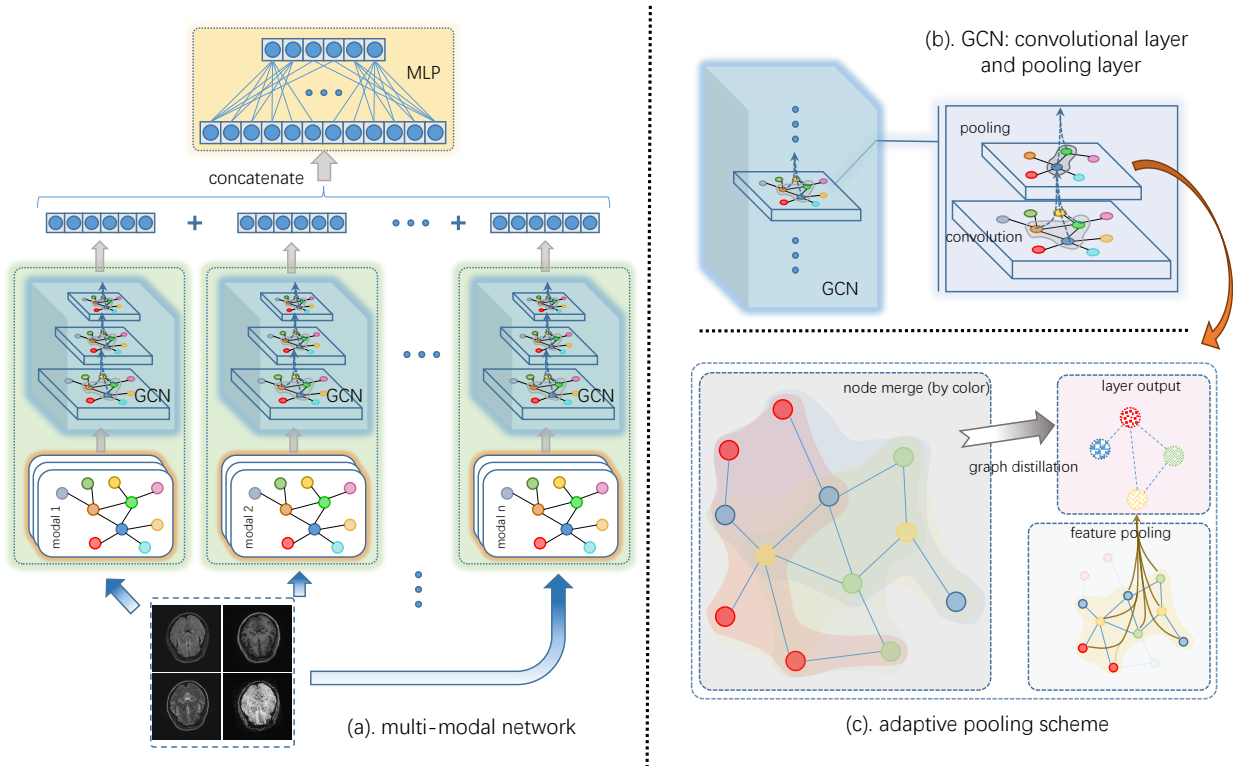


Figure 20: The proposed heterogeneous GCN for PD clinical scores prediction. (a) illustrates the entire structure, in which multi-modal brain networks are generated from MRIs, and sequentially processed by GCN and MLP; (b) depicts the stacked convolutional layer and pooling layer; (c) provides a detailed description of the pooling procedure, including node merge, graph distillation and feature pooling.

corresponding rows of brain network matrix as node features. We also propose a novel efficient adaptive pooling scheme, to learn data-driven pooling windows, construct reduced while structural-preserving graphs and aggregate pooled features.

Although graph convolution is a quite established technique, pooling on graphs is challenging in many senses. A major difficulty is the structural irregularity of graph data, compared to naive application scenarios such as images. For CNNs, typically pooling layer defines the operation on a window sliding along images with strides, which shrinks the input size and augments the receptive field of convolution. However, the window and the minified

graph are ambiguous due to the diverse topology. Another issue dedicated to our task is that the heterogeneity of brain networks calls for modal-specific designs. Last but not least, practical models should avoid potential computational inefficiency caused by complicated node operation. We prefer graph pooling sharing several appealing properties as CNN:

- **Locality:** The pooling windows aggregate local information from neighbor or related nodes.
- **Loyalty:** The reduced graph characterizes the structure of primary graph and data.
- **Likely:** The computation is efficient. Additionally, the data-driven pooling scheme should be end-to-end trainable, subject to deep learning principle.

To address these challenges, we imitate the process of CNN pooling, and decompose the graph pooling into two steps, node merge and graph distillation [154]. In the first step, nodes are clustered and features are computed accordingly; in the second, graph is reduced for follow-up graph convolution. Formally, the  $l$ -th pooling layer is defined as,

$$H_p^l = P^l H^l, \quad (5.14)$$

here  $H^l \in \mathcal{R}^{n^{l-1} \times k^l}$  is the output of graph convolutional layer  $l$ ,  $P^l \in \mathcal{R}^{n^l \times n^{l-1}}$  is a trainable pooling matrix,  $H_p^l \in \mathcal{R}^{n^l \times k^l}$  is the output of pooling layer, and  $n^l$  and  $k^l$  are the number of nodes and feature length, respectively. The reduced graph is defined as,

$$A^l = \arg \min_A \|A^{l-1} - P^l A P^{lT}\|_F + f(A, x), \quad (5.15)$$

here  $A^l$  is the graph matrix from layer  $l$ , and  $f(A, x)$  are defined to coincide in the naive regression objective, and  $\|\cdot\|_F$  is Frobenius norm. The first term can be derived by preserving the graph convolution consistency for pooled nodes,

$$U_P H_p^l \approx U P^l H^l, \quad (5.16)$$

here  $U_P$  and  $U$  are the Fourier basis of after and before pooling, respectively. Naturally, it can be interpreted as the eccentricity of the reduced graph to the graph at hand, and the second term considers the data fidelity. In principle, this pooling structure is defined using  $P^l$  and  $A^l$ , learned from both graph and data.

Now we delve into the details and discuss how the aforementioned concerns are resolved through some further consideration. We decompose the pooling matrix,

$$P^l = A_p^l A^{l-1}, \quad (5.17)$$

here  $A_p^l \in \mathcal{R}^{n^l \times n^{l-1}}$  is a sparse assigning matrix, and each row of  $A_p^l$  represents a cluster in the reduced graph. Each cluster aggregates the vicinity of assigned nodes; meanwhile, the co-occurrence of assigned (perhaps distant) nodes represent a high-level relation beyond neighboring. Therefore, *locality* is attained by sparsity regularization related to  $A_p^l$ . *Loyalty* is also maintained via compelling  $A^l$  to satisfy (5.15). Combining the above arguments boils down to the objective,

$$\mathcal{L} = \|x - \hat{x}\|_F + \lambda_1 \sum_l \|A^{l-1} - P^l A^l P^{lT}\|_F + \lambda_2 \sum_l \|A_p^l\|_1, \quad (5.18)$$

here the first term is the tedious regression loss,  $\lambda_1$  and  $\lambda_2$  are tunable parameters. This formulation only exerts slight computation burden to naive graph convolutional layer in the inference stage; the training is also an end-to-end routine on an integral structure and avoids some potential redundancy [154]. Jointly, these indicate the *likely* of the adaptive scheme in both training and computation. Finally, each modality is dealt with using individual GCN, with modal-specified graph and pooling setup, which leads to an intuitive explanation for the effectiveness of the proposed method, that *heterogeneity* is contained in the first stage, modal-fusion in the second.

## 5.2.3 Experiments

### 5.2.3.1 Data Description

We analyzed the data from PPMI (<http://www.ppmi-info.org>), which includes 145 healthy controls (HC) (mean *age* =  $66.70 \pm 10.95$ , 96 males) and 474 subjects with PD (mean *age* =  $67.33 \pm 9.33$ , 318 males). No significant differences was identified in age between HC and PD ( $P=0.5$ ). We utilized the diffusion-weighted MRI-derived structure connectome to predict several PD clinical scores, including the Montreal Cognitive Assessment (MoCA) Test, the Tremor Dominant (TD) scores and the Postural Instability and Gait Difficulty (PIGD)

scores, REM Sleep Behavior Disorder (RBD) scores, the Geriatric Depression Scale (GDS), and the University of Pennsylvania Smell Identification Test (UPSIT).

For each subject’s T1-weighted MRI, we applied ROBEX, a robust automated brain extraction program trained on manually “skull-stripped” MRI data [59], to remove the extra-cerebral tissue. These skull-stripped volumes were carefully examined and manually edited if needed. Anatomical scans were then underwent the standard FreeSurfer (V6.0, <http://surfer.nmr.mgh.harvard.edu/>) parcellation, based on which 84 cortical and subcortical ROIs are defined.

For each subject’s diffusion-weighted MRI, firstly *bet* and *eddy\_correct* functions in FSL (<http://www.fmrib.ox.ac.uk/fsl>) were applied to remove the non-brain tissue and correct for the possible distortions, and then the gradient table was adjusted correspondingly for each subject. In order to avoid the distortions at tissue-fluid interfaces, echo-planar induced susceptibility artifacts were corrected by elastically aligning skull-stripped b0 images to each subject’s T1 MRI using Advanced Normalization Tools (ANTs, <http://stnava.github.io/ANTs/>) with SyN algorithm. The resulted 3D deformation was then applied to the remaining diffusion-weighted volumes to generate the full preprocessed diffusion-weighted MRI data. Finally, based on the 84 ROIs derived from the T1 data, we reconstructed three brain structural graphs using three whole brain probabilistic tractography algorithms, including Orientation Distribution Function-based Hough voting [3] and PICo [98] as well as ball-and-sticks-based Probtrackx [11]. (please refer to [161] for more details). Each brain network was normalized by dividing the maximum values in the matrix to reduce the potential computation biases from the differences in scale and range from different tractography algorithms.

### 5.2.3.2 Experiment Settings

We compare the proposed method with several related methods: multivariate Ridge Regression(RR), Least Absolute Shrinkage and Selection Operator(LASSO), combined  $l_1$  and  $l_2$  norm (ElasticNet), Neural Networks (NN), and Convolutional Neural Network (CNN). For RR, LASSO and ElasticNet, we search the coefficient of the regularization term ranging from 0.001 to 100 and report the best results. For neural network, we use a two layer structure

with 100 hidden units and Relu activation function. For the proposed method, we use two layer GCN for each modality. The feature length for each layer is [16, 32] respectively, and the graph size after pooling is [32, 8]. An one-layer perceptron is used for regression. Both  $\lambda_1$  and  $\lambda_2$  are set to 0.001 in the objective. We use Adam optimizer [63] with a learning rate 0.001, and a batch size of 128. All clinical measures are normalized to [0, 1] by different tests. We reported the root mean square error (RMSE) and mean absolute value (MAE) on 5-fold cross validation as the evaluation metrics.

### 5.2.3.3 Results

We first present the comparison of the performance of the proposed method with multiple baselines, and the results are summarized in Table 14. On both metrics, we observe that the proposed method outperforms baselines consistently. For linear methods, sparse methods achieve better prediction accuracy compared to non-sparse methods, RR. Non-linear models, including NN, CNN, and the proposed model, also improve the prediction performances against linear models in general. The baseline deep methods, NN and CNN, have similar results. Particularly, the proposed method outperforms NN with much less parameters and has better performance with similar parameter size compared to CNN.

In Table 15 we also include the prediction performance of different combination of modals. Obviously, the network using single modal brain network yields much worse prediction, compared to multi-modal networks. The results also imply some interesting observations: particular subsets of involved modals may attain decent results, though the optimal combination require brutal search; the prediction yielded by the network integrating all modals, generally, is the best or at least comparable with the best. To this end it is fair to claim that multi-modality helps the prediction of PD clinical measures, and that more modals are always preferred.

### 5.2.4 Conclusion

In this paper, we propose a graph convolutional network to predict the PD clinical measures, using multi-modal brain networks. Particularly, we propose an adaptive pooling

Table 14: The comparison of the proposed method with baselines. For both metrics, smaller values indicate better results. The values are displayed as mean( $\mu$ )  $\pm$  standard deviation ( $\sigma$ ) from five tests. Bold font indicates the best performance.

Method	RMSE	MAE
RR	0.2382 $\pm$ 0.0069	0.1699 $\pm$ 0.0038
LASSO	0.2293 $\pm$ 0.0041	0.1613 $\pm$ 0.0033
ElasticNet	0.1969 $\pm$ 0.0011	0.1546 $\pm$ 0.0015
NN	0.1965 $\pm$ 0.0102	0.1544 $\pm$ 0.0057
CNN	0.1965 $\pm$ 0.0081	0.1545 $\pm$ 0.0049
Our method	<b>0.1922 <math>\pm</math> 0.0128</b>	<b>0.1455 <math>\pm</math> 0.0064</b>

Table 15: Predictions with different combination of modals. Values follow the instruction in Table. 14. Bold font indicates the best performance.

Modality	RMSE	MAE
Hough+Probtrackx+PICo	0.1922 $\pm$ 0.0128	<b>0.1455 <math>\pm</math> 0.0064</b>
Hough+Probtrackx	<b>0.1909 <math>\pm</math> 0.0111</b>	0.1458 $\pm$ 0.0060
Probtrackx+PICo	0.1948 $\pm$ 0.0123	0.1472 $\pm$ 0.0067
Hough+PICo	0.1927 $\pm$ 0.0126	0.1458 $\pm$ 0.0061
Hough	0.1941 $\pm$ 0.0124	0.1470 $\pm$ 0.0071
Probtrackx	0.1958 $\pm$ 0.0135	0.1488 $\pm$ 0.0070
PICo	0.1970 $\pm$ 0.0127	0.1501 $\pm$ 0.0068

scheme driven by both graph structure and brain data, which is efficient in computing and end-to-end training. The experiment results demonstrate that the proposed method attains state-of-the-art results compared to related baselines, and integrating multi-modal brain network is highly effective in the prediction task.

## 5.3 Unsupervised Multi-View Graph Representation Learning

### 5.3.1 Motivation

Human brain connectomes [19] are models of complex brain networks and can be derived from diverse experimental modalities and tractography algorithms. Large-scale brain connections convey important insights for understanding the underlying yet largely unknown mechanisms of many mental disorders [68, 82, 177, 22]. Nevertheless, the apparent characteristics of brain networks are profoundly influenced by the tractography algorithms. The designs of tractography algorithms, including tensor-based deterministic algorithms [3], probabilistic approaches [98], random forest [93] and Deep Neural Network (DNN) [106], and regularized methods guided by biologically plausible fascicle structures [5], are inspired by specific experimental questions [18], *e.g.*, different tractography algorithms are used for predicting or classifying neurodegenerative or neurodevelopmental conditions based on various brain abnormalities. For example, the selection and accuracy of the extracted fibers are different for different tractography algorithms, and the relevance of the extracted fiber bundles depend on the different tasks and questions being addressed. Therefore, it is elusive to decide a universally optimal modality of brain networks and associated processing pipeline for distinct diagnostic tasks [18, 137].

Multi-view methods can leverage the available information from diverse tractography algorithms simultaneously, and tentative studies have demonstrated that multi-modal brain networks can provide complementary viewpoints for the classification tasks, *e.g.*, multi-view graph convolutional network [167] is found to have state-of-the-art performance in classifying Parkinson’s disease (PD) status. However, previous multi-view methods have two restrictions regarding general prediction tasks of neurodegenerative conditions. First, many methods are designed for some specific tasks. If one want to tailor these methods to other tasks, it is necessary to carefully tune the hyperparameters. Second, though some methods learn representations from multi-view brain networks, the learning is guided by some predefined prediction tasks, which may introduce bias to overemphasize a particular modality. As such, the learned embeddings cannot represent multi-modal brain networks comprehensively, and

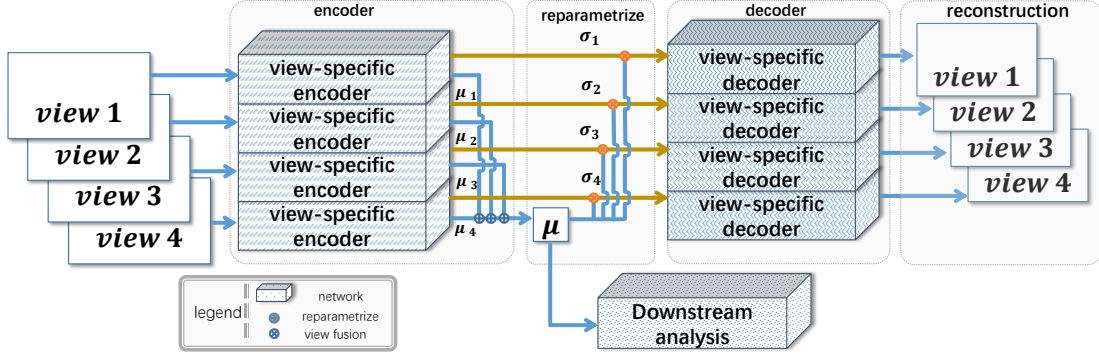


Figure 21: The structure of the proposed method. Each view uses an independent VGAE to learn a unified  $\mu$ , while the  $\sigma$  is different.

their application to the related analysis in a broader scope is potentially constrained.

To address these problem, we propose to learn unified representations from multi-modal brain networks via unsupervised learning techniques. To extend the generalization ability of the learned representations to different downstream analysis, the representations shall be of *disentanglement* and *proportionality* concerning different modalities. Here, disentanglement refers to the representations encoding salient attributes of data explicitly, which can help the analysis of the prediction tasks and the modalities. Proportionality refers to a balanced contribution to the representations of each modality, which avoids the potential bias on specific modalities. In other words, in our approach the learned representations can fairly convey the information from different modalities and can be exploited by various downstream analysis. More specifically, in this paper we propose a multi-view graph auto-encoder to learn the disentangled graph embeddings from brain networks. We formulate the proportionality-awareness in multi-view representation learning as a network scheduling problem via an analogy between training deep networks and the graph flow problems. The experimental results demonstrate the effectiveness of the proposed method.



### 5.3.2 Methodology

The proposed method is illustrated in Fig. 21. For each view, a Variational Graph Auto-encoder (VGAE) [65] is exploited. Let  $G^{(v)}$  denote the brain networks of the  $v^{th}$  view,  $f^{(v)}$  and  $g^{(v)}$  the corresponding encoder and decoder,  $[\mu^{(v)}|\sigma^{(v)}] = f(G^{(v)})$  is the estimated mean and variance of the encoder. The unified representations are computed by max-out the stacked  $\mu^{(v)}$  by the position, which can be denoted as  $\mu = \text{maxpool1d}([\mu^{(v)}])$ . The reparameterization for the  $v^{th}$  view is then computed using  $\mu$  and  $\sigma^{(v)}$ .  $\mu \in \mathbb{R}^k$  is also used as the embeddings. According to the structure of VGAE,  $\sigma^{(v)} \in \mathbb{R}^k$ . Besides the view-wise VGAE loss, we push  $\mu$  and  $\mu^{(v)}$  to be close so that the learned embeddings for different views are consistent. The disentanglement of the representations is acquired via introduce the  $\beta$ -VAE loss [56]. Disentangled representations are compact and interpretable [14]. The objective for our multi-view GVAE is:

$$\mathcal{L} = \sum_{v \in \mathcal{V}} \mathbb{B} \left( \log \left( P(\tilde{G}^{(v)}) \right) \right) + \beta KL \left( P(z^{(v)}) | \mathcal{N}(0, 1) \right) + \lambda (\mu^{(v)} - \mu)^2 \quad (5.19)$$

here the first term is the reconstruction loss, the second is the Kullback-Leibler divergence, and the last is the multi-view consistency.

As aforementioned, the representations shall also be fair to different views. In the above auto-encoder framework, the decoder is used for evaluating the vividness of the learned representations. However, for multi-view data, the reconstruction for different views is not necessarily equally accurate. When the imbalance occurs, some views are less included in the learned representations. To address this problem, we consider to learn fair representations regarding different views, which indicates *the view-wise loss in (5.19) is close to each other*. Such fairness, referred to as *proportionality*, can be achieved via an alternative training routine of the above model. We will formulate an analog between flow network problem and the training of multi-view model in the following. Based on the formulation, we design a scheduling algorithm to satisfy the proportionality requirement.

**Training Multi-view Network: a Flow Network Perspective** Directed Acyclic Graph (DAG) is an important tool in graphical models [57]. It is also exploited to express network

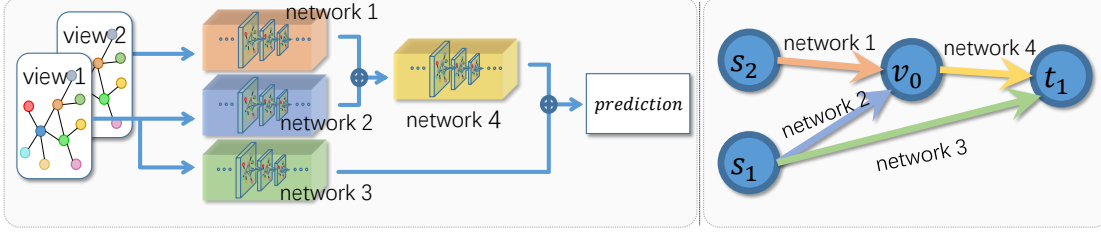


Figure 22: Left: a simple DNN. Right: the corresponding DAG. Each edge represents a network, and each node denote an intermediate representation.

structures by many popular deep learning frameworks [99, 1]. Inspired by this idea, we make an analogy between training the deep network and the flow network problems.

In Fig. 22, we illustrate an example for multi-view learning. To simplify the elaboration, we consider a structure taking two views  $s_1$  and  $s_2$ , as inputs. The network consists of four sub-networks, each corresponding to one edge in the DAG.  $v_0$  is a fused hidden representation, and  $t_1$  is the prediction. For multiple inputs,  $\oplus$  denotes the fusion operation for the outputs of multiple sub-networks, and it can either a weighted summation or concatenation. Consider a network trained after  $t$  steps using gradient based method. In the  $t + 1$  step, we can define the flow  $d_{i,j}$  from predecessor  $i$  to successor  $j$  as  $d_{i,j} = \Delta \mathcal{L} \left( f_j^{(t+1)}(h_i^{(t+1)}, \mathcal{H}_{j \setminus i}^{(t)}) \right)$ , here  $\mathcal{L}$  is an objective defined on the targets, and  $\Delta \mathcal{L}$  denotes the loss difference between step  $t + 1$  and  $t$ . Let  $\mathcal{P}_{ij}$  represent the set of all paths from sources to targets containing  $e_{i,j}$ .  $f_j^{(t+1)}$  refers to the network to compute the final outputs with all paths in  $\mathcal{P}_{ij}$  updated.  $\mathcal{P}_{ij}$  can be defined on the node  $i$  and a set  $\mathcal{H}_{j \setminus i}$ . Here  $\mathcal{H}_{j \setminus i}$  denotes any cut set containing node  $j$  that separate sources and targets, and  $\mathcal{H}_{j \setminus i}$  does not include any node in  $\mathcal{P}_{ij}$  except  $j$ .

Our definition satisfies the flow conservation, which states that if a node is neither a source or a target, its net flow shall be 0. For a node  $j$  with multiple incoming flow, the fusion operation is defined as  $\mathbf{h}_j = \sum_{i \in \mathcal{P}_j} \mathbf{P}_{ij} \mathbf{W}_{ij} f_{ij}(\mathbf{h}_i)$ , here  $\mathcal{P}_j$  is the predecessor set of node  $j$ ,  $f_{ij}$  is the sub-network between node  $i$  and  $j$ . For different fusion operations,  $\mathbf{P}_i$  and  $\mathbf{W}_i$  take different forms. For example, when both  $\mathbf{P}_i$  and  $\mathbf{W}_i$  are the identity matrices, the fusion is by summation; if  $\mathbf{W}_i$  is the augmented matrix  $(\mathbf{I}_i | \mathbf{0})$ , fusion by concatenation is feasible

by setting  $\mathbf{P}_i$  as the corresponding permutation matrix. For a node with multiple outgoing flow, the output is equally distributed. We abuse the notation  $\mathcal{H}_j \equiv \mathcal{H}_{j \setminus i} \cup \{i\}$ . Consider a fixed given cut  $\mathcal{H}_j$  for node  $j$ , we can induce two additional cuts:  $\mathcal{H}_{\mathcal{P}_j}$ , which excludes  $j$  and include all its predecessors; and  $\mathcal{H}_{\mathcal{S}_j}$ , which excludes  $j$  and include all its successors. Under the updating rule of backward propagation, the incoming flow with respect to node  $j$  is,

$$\sum_{i \in \mathcal{P}_j} d_{i,j} \approx \frac{\partial \mathcal{L}}{\partial f_j} \sum_{i \in \mathcal{P}_j} \mathbf{P}_{ij} \mathbf{W}_{ij} \frac{\partial f_j}{\partial \mathbf{h}_i} d\mathbf{h}_i = \frac{\partial \mathcal{L}}{\partial f_j} \frac{\partial f_j}{\partial \mathbf{h}_j} d\mathbf{h}_j, \quad (5.20)$$

the above equation follows because the partial differential is 0 except  $d\mathbf{h}_i$  and  $d\mathbf{h}_j$  term. Similarly, the outgoing flow is,

$$\sum_{k \in \mathcal{S}_j} d_{j,k} \approx \sum_{k \in \mathcal{S}_j} \frac{\partial \mathcal{L}}{\partial f_k} \mathbf{P}_{jk} \mathbf{W}_{jk} \frac{\partial f_k}{\partial \mathbf{h}_j} d\mathbf{h}_j = \frac{\partial \mathcal{L}}{\partial f_j} \frac{\partial f_j}{\partial \mathbf{h}_j} d\mathbf{h}_j, \quad (5.21)$$

(5.20) and (5.21) are bridged by the change in  $h_j$ , which ensures the net flow to be 0.

If we extend the above analogy to the accumulative case, the flow is defined to be the loss decrease with respect to the particular structure represented by  $i \rightarrow j$ . Noteworthy, it is not the pure contribution of  $i \rightarrow j$ . Rather, it is more of the quantification of the total loss decrease of the particular structure, as the definition considers both the upstream and downstream computation of the entire network. The empirical loss is related to the generalization bound of the learned representations concerning downstream tasks. As such, the accumulated flow can be interpreted as the amount of information learned from each view informally. Based on this analogy, we define that the proportionality is achieved if the view-wise flow, *i.e.* the accumulated  $\sum_{k \in \mathcal{S}_j} d_{j,k}$  for some view  $j$ , is balanced.

**Alternative Training Routine with Proportionality Awareness** Conventionally, the proportionality concerning different views can be written as a constrained optimization problem, and a standard training routine is based on SGD. From the flow perspective, the proportional training can be interpreted as multiple views competing for the updating resources in the backward propagation, which is a network scheduling algorithm. More specifically, during the training, the accumulated flow is continuously updated, which reflexes the dynamic of loss decrease and the generalization ability. A proportional representation is then equivalent to a balanced flow avoiding the overload of some specific path.

---

Round-Robin
<b>Input:</b> $v$ views, max epoch $e$
<b>Output:</b> model $f$
1 Initialize $f$ .
2 <b>repeat</b>
3     <b>for</b> $i \leftarrow 1$ <b>to</b> $v$ <b>do</b>
4           Optimize (5.19) <i>w.r.t.</i> view $j$ .
5 <b>until</b> <i>max epoch</i> ;

---



---

Proportionality
<b>Input:</b> $v$ views, max epoch $e$
<b>Output:</b> model $f$
1 Initialize $f$ .
2 <b>repeat</b>
3       Compute priority <i>w.r.t.</i> (5.24)
4       Optimize (5.19) <i>w.r.t.</i> view $j$
with the highest priority.
5 <b>until</b> <i>max epoch</i> ;

---

In detail, we define the total flow as the loss decrease. When the learning rate is small enough, the summation of view-wise SGD update is equivalent to a *round-robin* update with respect to each view. Here, the objective associated with each view is optimized in a predefined turn. To avoid a specific view taking up too much updating resources, we can maximize the total flow of the network while allowing the minimal level of service for all views via introducing a competing mechanism for each view to occupy the update based on the estimated flow. We refer to this method as *proportionality*. The updating priority of each view is based on the current loss decrease and the historical cumulative loss decrease. Assume the loss decrease of view  $i$  at update  $t$  can be foreseen as  $r_{i,t}$ . The throughput of view  $i$  is defined as historical cumulative loss decrease at step  $t$ :

$$\theta_{i,t} = \theta_{i,0} + \sum_{l=1}^t \frac{r_{i,l} I_{i,l}}{t} = \frac{n-1}{n} \theta_{i,t-1} + \frac{1}{n} r_{i,t-1} I_{i,t-1}, \quad (5.22)$$

where  $I_{i,l}$  is an indicator.  $I_{i,l} = 1$  if the  $l^{\text{th}}$  update is conducted on view  $i$ , and 0 otherwise. Based on (5.22), the priority  $p_{i,t}$  for view  $i$  can be defined, and the  $t+1$  update is then applied to the view with the highest priority:

$$\arg \max_{i \in \mathcal{V}} \{p_{i,t}\}, \quad p_{i,t} = \frac{r_{i,t+1}}{\epsilon + \theta_{i,t}} \quad (5.23)$$

where  $\epsilon$  is a small positive number for computational stability. Notably, the above algorithms is not immediately applicable to our formulation, as that  $r_{i,t}$  is not pre-assigned as in standard

proportionally fairness algorithms. Instead, the values are only known after the update is finished. Thus, we propose a compensation update method: at the beginning, we use one round robin update and compute initial  $r_{i,0}$ . In the following steps we use proportionally fairness algorithm, but computing the priority using the loss decrease from the last applied update:

$$\arg \max_{i \leq v} \left\{ \frac{r_{i,t_i}}{d_i + \theta_{i,t}} \right\}, \quad t_i = \max l, \quad s.t. \quad l \leq t, \quad I_{i,l} = 1, \quad (5.24)$$

The proportionality and convergence of our scheduling algorithm are guaranteed under some weak conditions, and the analysis can be found in [70].

### 5.3.3 Experimental Results

In this experiments we use three datasets, including the data from the Alzheimer’s Disease Neuroimaging Initiative (ADNI) and National Alzheimer’s Coordinating Center (NACC), and the Parkinson Progression Marker Initiative (PPMI). The preprocessed ADNI brain networks [136] include 51 healthy controls (HC) (mean age= $69.69 \pm 10.27$ , 29 males), 112 people with Mild Cognitive Impairment (MCI) (mean age= $71.68 \pm 9.89$ , 41 males) and 39 individuals with AD (mean age= $75.56 \pm 8.99$ , 14 males). The similarly preprocessed NACC brain networks [135] include 329 HCs (mean age= $60.96 \pm 8.96$ , 107 males), 57 with MCI (mean age= $73.60 \pm 7.93$ , 38 males), and 54 AD patients (mean age =  $72.02 \pm 10.41$ , 32 males). The similarly preprocessed PPMI brain networks [180, 181] includes 145 HC (mean age =  $66.70 \pm 10.95$ , 96 males) and 474 subjects with PD (mean age= $67.33 \pm 9.33$ , 318 males). Nine different views are reconstructed using T-FACT, T-RK2, T-TL, T-SL, O-FACT and O-RK2, Probt, Hough, and PICo (Please refer to [161] for more details on the brain network reconstruction). We use a modified network structure based on graph variational auto-encoder. The view-wise graph is the averaged brain connectome, and the node features are the corresponding row for each brain connectome. We set  $\beta = 4$  recommended by  $\beta$ -VAE [56]. The performance is not sensitive to  $\lambda$ , and we set it to 0.001. In the encoder, we use three graph convolutional layers for  $\mu$  and  $\sigma$  respectively. The first two layers are shared, both with 64 hidden units. The embedding length is 32. The encoder are limited in layers

due to the potential over-smoothing for graph convolutional layers. Our model is trained 100 epochs using ADAM with batch size 32 and learning rate 0.0001.

**Evaluating the Proposed Method in Down-streaming Analysis:** We compare our approach with related baselines on several classification and regression tasks. The ablation study is also included.

Table 16 summarizes the classification results. For ADNI and NACC, we predict the HC and AD. For PPMI we predict HC and PD. For multi-view predictions, we include principal component analysis (PCA), multi-view non-negative matrix factorization (MVNMF) [78], co-regularized spectral clustering (MVSC) [69] and Deep Metric Graph Convolutional Network (DMGCN) [68]. We use the aforementioned methods to learn the representations, and then exploit two off-the-shelves methods, sparse logistic regression, and random forest to make the final prediction. We report AUC on 5-fold cross-validation. To make the comparison self-contained, single view results are also included. For the ablation study, in *propose-I* neither disentanglement nor proportionality is considered, and in *proposed-II* the disentanglement is considered. The full approach is *proposed\**. We omit more single-view ablation study in the experiments because our objective is designed for multi-view data. Of note, integrating multi-view data is also shown to be beneficial for brain network analysis [180]. From the results, we find that the prediction ability of different views with respect to different tasks are complicated, and heavily coupled with the algorithms. Multi-view methods, generally, can improve the prediction ability. However, the advantage of multi-view data is intriguing and needs careful examination. The proposed method have good performances and are robust with respect to different tasks. And at last, the ablation study demonstrates that the performance can be improved through considering disentanglement and proportionality.

Table 17 summarizes the regression results. We use the learned representation to predict several clinical scores, including the Tremor Dominant scores (TD), the University of Pennsylvania Smell Identification Test (UPSIT), and the Montreal Cognitive Assessment Test (MoCA). Mean squared error (MSE) is used as the metric evaluating the prediction. All scores are normalized to  $[0, 1]$ . The results show that the prediction is more complicated with respect to the particular medical scores and views. Similarly, we can observe the advantage of utilizing multi-view data and the robust and superior prediction abilities of our approach.

Table 16: The comparison on classification tasks.

Sparse Logistic Regression				
		ADNI	NACC	PPMI
Single View	FSL	$0.7786 \pm 0.0976$	$0.7669 \pm 0.0799$	<b><math>0.6597 \pm 0.0584</math></b>
	PICo	$0.7615 \pm 0.1408$	$0.7119 \pm 0.1103$	$0.6065 \pm 0.0486$
	T-FACT	$0.7451 \pm 0.0379$	$0.6581 \pm 0.0411$	$0.5850 \pm 0.0433$
	O-FACT	$0.7278 \pm 0.1066$	$0.7094 \pm 0.0866$	$0.5921 \pm 0.0353$
	ODF-Rk2	$0.7568 \pm 0.0821$	$0.6890 \pm 0.0366$	$0.5942 \pm 0.0331$
	T-RK2	$0.7276 \pm 0.0797$	$0.7281 \pm 0.0674$	$0.5921 \pm 0.0353$
	T-SL	$0.7402 \pm 0.1371$	$0.6582 \pm 0.0785$	$0.5884 \pm 0.0389$
	T-TL	$0.6875 \pm 0.0682$	$0.7358 \pm 0.0799$	$0.5851 \pm 0.0423$
	Hough	$0.7559 \pm 0.0780$	$0.7271 \pm 0.0549$	$0.5536 \pm 0.0391$
Multi View	all views	$0.7966 \pm 0.0904$	$0.7301 \pm 0.1325$	$0.5716 \pm 0.0378$
	MVNMF	$0.8149 \pm 0.0550$	$0.7685 \pm 0.0958$	$0.6104 \pm 0.0332$
	MVSC	$0.8203 \pm 0.0791$	$0.7595 \pm 0.1013$	$0.6205 \pm 0.0373$
	DMGCN	$0.8058 \pm 0.1006$	$0.7557 \pm 0.0898$	$0.6141 \pm 0.0707$
	Proposed-I	$0.8074 \pm 0.0493$	$0.7491 \pm 0.0897$	$0.6122 \pm 0.0442$
	Proposed-II	$0.8185 \pm 0.0770$	$0.7549 \pm 0.0790$	$0.6240 \pm 0.0234$
	proposed*	<b><math>0.8278 \pm 0.1537</math></b>	<b><math>0.8090 \pm 0.1472</math></b>	$0.6250 \pm 0.0472$
Random Forest				
		ADNI	NACC	PPMI
Single View	FSL	$0.8124 \pm 0.0455$	$0.3737 \pm 0.7065$	$0.5753 \pm 0.0255$
	PICo	$0.7838 \pm 0.1067$	$0.1588 \pm 0.9463$	$0.5475 \pm 0.0244$
	T-FACT	$0.8383 \pm 0.0483$	$0.7029 \pm 0.1184$	$0.5654 \pm 0.0331$
	O-fact	$0.7817 \pm 0.1512$	$0.3789 \pm 0.6903$	$0.5478 \pm 0.0228$
	O-RK2	$0.7617 \pm 0.1087$	$0.7879 \pm 0.1333$	$0.5566 \pm 0.0382$
	T-RK2	$0.7764 \pm 0.1275$	$0.7029 \pm 0.1333$	$0.5486 \pm 0.0361$
	T-SL	$0.8148 \pm 0.0587$	$0.7235 \pm 0.1163$	$0.5386 \pm 0.0347$
	T-TL	$0.7695 \pm 0.0862$	$0.7009 \pm 0.1164$	$0.5411 \pm 0.0410$
	Hough	$0.8368 \pm 0.0671$	$0.7011 \pm 0.1797$	$0.5276 \pm 0.0344$
Multi View	all views	$0.8560 \pm 0.0574$	$0.7615 \pm 0.1053$	$0.5743 \pm 0.0464$
	MVNMF	$0.8826 \pm 0.0830$	$0.8317 \pm 0.1561$	$0.5659 \pm 0.0528$
	MVSC	$0.8827 \pm 0.0457$	$0.7997 \pm 0.1435$	$0.5753 \pm 0.0348$
	DMGCN	$0.8862 \pm 0.0503$	$0.8307 \pm 0.1493$	$0.5683 \pm 0.0323$
	Proposed-I	$0.8578 \pm 0.0516$	$0.7919 \pm 0.0725$	$0.5590 \pm 0.0250$
	Proposed-II	$0.8678 \pm 0.0573$	$0.8327 \pm 0.0988$	$0.5699 \pm 0.0382$
	Proposed*	<b><math>0.8946 \pm 0.0510</math></b>	<b><math>0.8359 \pm 0.1321</math></b>	<b><math>0.5814 \pm 0.0274</math></b>

Table 17: The comparison on regression tasks.

		TD	UPSIT	MoCA
Single View	FSL	$0.0749 \pm 0.0167$	$0.0794 \pm 0.0054$	$0.0381 \pm 0.0033$
	PICo	$0.0714 \pm 0.0078$	$0.0983 \pm 0.0101$	$0.0394 \pm 0.0027$
	T-FACT	$0.0404 \pm 0.0037$	$0.0545 \pm 0.0043$	$0.0210 \pm 0.0021$
	O-FACT	$0.0410 \pm 0.0018$	$0.0508 \pm 0.0066$	$0.0208 \pm 0.0036$
	O-RK2	$0.0428 \pm 0.0079$	$0.0503 \pm 0.0015$	$0.0208 \pm 0.0027$
	T-RK2	$0.0441 \pm 0.0034$	$0.0500 \pm 0.0051$	$0.0212 \pm 0.0025$
	T-SL	$0.0427 \pm 0.0012$	$0.0512 \pm 0.0058$	$0.0212 \pm 0.0017$
	T-TL	$0.0406 \pm 0.0059$	$0.0517 \pm 0.0019$	$0.0210 \pm 0.0027$
	Hough	$0.0434 \pm 0.0036$	$0.0495 \pm 0.0058$	$0.0225 \pm 0.0044$
Multi View	all views	$0.0414 \pm 0.0045$	$0.0524 \pm 0.0034$	$0.0227 \pm 0.0074$
	MVNMF	$0.0378 \pm 0.0122$	$0.0507 \pm 0.0041$	$0.0207 \pm 0.0030$
	MVSC	$0.0355 \pm 0.0047$	$0.0499 \pm 0.0046$	<b><math>0.0199 \pm 0.0013</math></b>
	DMGCN	$0.0365 \pm 0.0071$	$0.0487 \pm 0.0085$	$0.0202 \pm 0.0015$
	Proposed-I	$0.0358 \pm 0.0024$	$0.0501 \pm 0.0022$	$0.0209 \pm 0.0019$
	Proposed-II	$0.0361 \pm 0.0035$	$0.0492 \pm 0.0033$	$0.0200 \pm 0.0022$
	Proposed*	<b><math>0.0351 \pm 0.0059</math></b>	<b><math>0.0484 \pm 0.0044</math></b>	<b><math>0.0199 \pm 0.0022</math></b>

**Evaluating the Proportionality during Training:** In this section, we demonstrate the proposed method can achieve proportionality using the proposed training scheduling method. Figure 24 illustrates the training loss of the proposed deep network against epochs, and the shaded area represents the variance regarding different views. From the results, we can observe that the proposed method effectively reduces the variance during training, which indicates the learned representations proportionally represent different modalities of brain networks. The results also show the training routine aware of proportionality converges slightly slower than the standard training routine. However, with moderate epochs their performance difference is negligible.

**Discussions:** There some works applying the fairness principle on brain analysis [90]. Our method is designed for representation learning for multi-view brain connectomes, particularly focusing on the disentangled and proportional property (which is related to algorithmic fairness) for the learned embeddings. Our experimental results demonstrate that the proposed method can be applied to various downstream works. As such, it is of potential to apply our



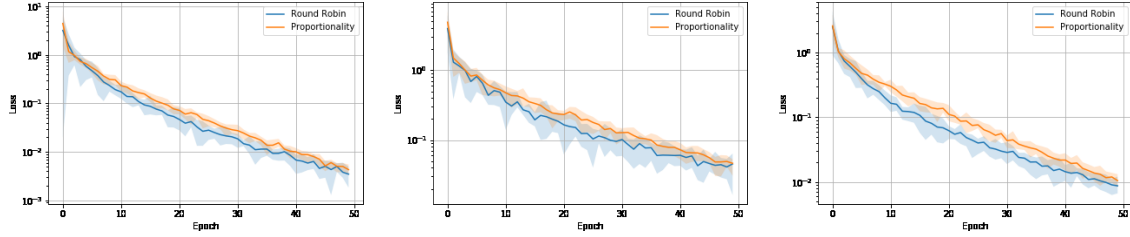


Figure 24: Left to right: ADNI, NACC, PPMI.

method to broader applications, including generating a refined connectome matrix,

### 5.3.4 Conclusion

In this paper, we propose an unsupervised method to learn unified graph embeddings for multi-view brain networks. We design a multi-view graph variational auto-encoder to learn the representations with disentanglement and proportionality. The experimental results demonstrate that the learned representations can be effectively used by various downstream tasks.

## 6.0 Conclusion

In this dissertation, we propose several new representation-learning algorithms to address the challenges of structural big data mining. Firstly, we propose a framework employing second-order graph neural networks, which usually learn a less stiff transformation than the first-order counterpart. Our method can also be viewed as a coupled first-order model, which is easy to implement. We propose a semi-model-agnostic method based on our model to enhance the prediction explanation using high-order information. We construct an analog between continuous GNNs and some famous partial differential equations and discuss some properties of the first and second-order models. Secondly, we learn structural embeddings in which the proximity is characterized by 1-Wasserstein distance. We propose a distributionally robust self-supervised graph neural network framework to learn the representations. More specifically, in our method, the embeddings are computed based on subgraphs centering at the node of interest and represent both the node of interest and its neighbors, which better preserves the local structure of nodes. To make our model end-to-end trainable, we adopt a deep implicit layer to compute the Wasserstein distance, which can be formulated as a differentiable convex optimization problem. Meanwhile, our distributionally robust formulation explicitly constrains the maximal diversity for matched queries and keys. As such, our model is insensitive to the data distributions and has better generalization abilities. Lastly, We propose a faster training method for GCN. The bottleneck in training GCN is introduced by both the network depth and the sampling behavior. Instead of improving the sampling, we focus on “reducing” the depth via reusing stale gradients.

We also design several representation learning methods to show their effectiveness on application problems. Firstly, we learn the shared graph structure to characterize the graph distribution using GNNs based on a small-world model. In detail, we showed that GCN without given graph structure is applicable, by using a naive complete graph. Meanwhile, a method was proposed to learn the graph structure from data to improve the performance of GCN, by generating random graph with small-world property for model training. Secondly, we propose a heterogeneous GCN for multi-view networks. An adaptive pooling scheme, driven

by both graph structure and network patterns, is also proposed, which is beneficial from gathering local information, yielding a faithful graph with smaller size, and enjoying efficiency in both computation and training. And thirdly, we propose to learn unified representations from multi-modal brain networks via unsupervised learning techniques. To extend the generalization ability of the learned representations to different downstream analysis, the representations shall be of disentanglement and proportionality concerning different modalities. In our approach the learned representations can fairly convey the information from different modalities and can be exploited by various downstream analysis.

## Bibliography

- [1] Martín Abadi, , et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, pages 265–283, 2016.
- [2] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [3] Iman Aganj et al. A hough transform global probabilistic approach to multiple-subject diffusion mri tractography. *Medical image analysis*, 15(4):414–425, 2011.
- [4] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and Zico Kolter. Differentiable convex optimization layers. *arXiv preprint arXiv:1910.12430*, 2019.
- [5] Farzane Aminmansour et al. Learning macroscopic brain connectomes via group-sparse factorization. In *Advances in Neural Information Processing Systems*, pages 8847–8857, 2019.
- [6] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.
- [7] James Atwood et al. Diffusion-convolutional neural networks. In *NeurIPS*, pages 1993–2001, 2016.
- [8] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54, 2006.
- [9] Shane Barratt. On the differentiability of the solution to convex optimization problems. *arXiv preprint arXiv:1804.05098*, 2018.
- [10] Danielle Smith Bassett et al. Small-world brain networks. *The neuroscientist*, 12(6):512–523, 2006.

- [11] Timothy EJ Behrens et al. Probabilistic diffusion tractography with multiple fibre orientations: What can we gain? *Neuroimage*, 34(1):144–155, 2007.
- [12] Mikhail Belkin et al. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NeurIPS*, pages 585–591, 2002.
- [13] Mikhail Belkin et al. Towards a theoretical foundation for laplacian-based manifold methods. *Journal of Computer and System Sciences*, 74(8):1289–1308, 2008.
- [14] Yoshua Bengio et al. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.
- [15] Austin R Benson, David F Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.
- [16] Léon Bottou et al. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.
- [17] Joan Bruna et al. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.
- [18] Ed Bullmore et al. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature reviews neuroscience*, 10(3):186, 2009.
- [19] Edward T Bullmore and Danielle S Bassett. Brain graphs: graphical models of the human brain connectome. *Annual review of clinical psychology*, 7:113–140, 2011.
- [20] Liu Cao, Longxu Jin, Hongjiang Tao, Guoning Li, Zhuang Zhuang, and Yanfu Zhang. Multi-focus image fusion based on spatial frequency in discrete cosine transform domain. *IEEE signal processing letters*, 22(2):220–224, 2014.
- [21] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900, 2015.
- [22] Chelsea Caspell-Garcia et al. Multiple modality biomarker prediction of cognitive impairment in prospectively followed de novo parkinson disease. *PLoS One*, 12(5):e0175674, 2017.

- [23] Jianfei Chen et al. Stochastic training of graph convolutional networks with variance reduction. In *International conference on machine learning*, pages 941–949, 2018.
- [24] Jie Chen et al. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.
- [25] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.
- [26] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.
- [27] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020.
- [28] R Cameron Craddock et al. Disease state prediction from resting state functional connectivity. *Magnetic Resonance in Medicine*, 62(6):1619–1628, 2009.
- [29] Miles Cranmer, Sam Greydanus, et al. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.
- [30] Michaël Defferrard et al. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, pages 3844–3852, 2016.
- [31] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multi-variate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4027–4035, 2021.
- [32] Zhiwei Deng, Megha Nawhal, Lili Meng, and Greg Mori. Continuous graph flow. *arXiv preprint arXiv:1908.02436*, 2019.
- [33] Inderjit S Dhillon et al. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE TPAMI*, 29(11), 2007.
- [34] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th ACM SIGKDD*

- International Conference on Knowledge Discovery & Data Mining*, pages 1320–1329, 2018.
- [35] Simon S Du et al. Gradient descent finds global minima of deep neural networks. *arXiv preprint arXiv:1811.03804*, 2018.
- [36] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. *arXiv preprint arXiv:1904.01681*, 2019.
- [37] David K Duvenaud et al. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.
- [38] David K Duvenaud, Dougal Maclaurin, et al. Convolutional networks on graphs for learning molecular fingerprints. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, pages 2224–2232. Curran Associates, Inc., 2015.
- [39] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on Learning Theory*, pages 907–940, 2016.
- [40] Lawrence C Evans. Partial differential equations. *Graduate studies in mathematics*, 19(2), 1998.
- [41] Daniel R Figueiredo, Leonardo Filipe Rodrigues Ribeiro, and Pedro HP Saverese. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada*, pages 13–17, 2017.
- [42] Alex Fornito et al. Graph analysis of the human connectome: promise, progress, and pitfalls. *Neuroimage*, 80:426–444, 2013.
- [43] Igor Fortel, Mitchell Butler, Laura E Korthauer, Liang Zhan, Olusola Ajilore, Ira Driscoll, Anastasios Sidiropoulos, Yanfu Zhang, Lei Guo, Heng Huang, et al. Brain dynamics through the lens of statistical mechanics by unifying structure and function. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part V 22*, pages 503–511. Springer, 2019.

- [44] Hongchang Gao et al. Identifying connectome module patterns via new balanced multi-graph normalized cut. In *MICCAI*, pages 169–176, 2015.
- [45] Hongchang Gao and Heng Huang. Deep attributed network embedding. In *Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.
- [46] Shangqian Gao, Feihu Huang, Yanfu Zhang, and Heng Huang. Disentangled differentiable network pruning. In *European Conference on Computer Vision*, pages 328–345. Springer, 2022.
- [47] Justin Gilmer, Samuel S Schoenholz, et al. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR, 2017.
- [48] Sam Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *arXiv preprint arXiv:1906.01563*, 2019.
- [49] Aditya Grover et al. node2vec: Scalable feature learning for networks. In *ACM SIGKDD*, pages 855–864. ACM, 2016.
- [50] Will Hamilton et al. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.
- [51] William L Hamilton et al. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.
- [52] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pages 4116–4126. PMLR, 2020.
- [53] Kaiming He et al. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [54] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.



- [55] Keith Henderson et al. Rolx: structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1231–1239, 2012.
- [56] Irina Higgins et al. beta-vae: Learning basic visual concepts with a constrained variational framework. *Iclr*, 2(5):6, 2017.
- [57] Feihu Huang and Songcan Chen. Learning dynamic conditional gaussian graphical models. *IEEE Transactions on Knowledge and Data Engineering*, 30(4):703–716, 2017.
- [58] Wenbing Huang et al. Adaptive sampling towards fast graph representation learning. In *Advances in Neural Information Processing Systems*, pages 4563–4572, 2018.
- [59] Juan Eugenio Iglesias et al. Robust brain extraction across datasets and comparison with publicly available methods. *IEEE transactions on medical imaging*, 30(9):1617–1634, 2011.
- [60] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [61] Nal Kalchbrenner et al. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.
- [62] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.
- [63] Diederik P Kingma et al. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [64] Thomas N Kipf et al. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [65] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.
- [66] Steven G Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2012.

- [67] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [68] Sofia Ira Ktena et al. Distance metric learning using graph convolutional networks: Application to functional brain networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 469–477. Springer, 2017.
- [69] Abhishek Kumar, Piyush Rai, and Hal Daume. Co-regularized multi-view spectral clustering. In *Advances in neural information processing systems*, pages 1413–1421, 2011.
- [70] Harold J Kushner et al. Convergence of proportional-fair sharing algorithms under general conditions. *IEEE T. on Wireless Communications*, 3(4):1250–1259, 2004.
- [71] Denis Le Bihan et al. Diffusion tensor imaging: concepts and applications. *Journal of Magnetic Resonance Imaging*, 13(4):534–546, 2001.
- [72] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [73] Yann LeCun et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.
- [74] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International conference on machine learning*, pages 3734–3743. PMLR, 2019.
- [75] Cheng Li et al. From which world is your graph. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1469–1479. Curran Associates, Inc., 2017.
- [76] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9267–9276, 2019.
- [77] Fan Liu, Zhiyong Cheng, Lei Zhu, Zan Gao, and Liqiang Nie. Interest-aware message-passing gcn for recommendation. In *Proceedings of the Web Conference 2021*, pages 1296–1305, 2021.

- [78] Jialu Liu, Chi Wang, Jing Gao, and Jiawei Han. Multi-view clustering via joint nonnegative matrix factorization. In *SIAM International Conference on Data Mining*, pages 252–260. SIAM, 2013.
- [79] Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. *arXiv preprint arXiv:1808.09602*, 2018.
- [80] Dijun Luo et al. New probabilistic multi-graph decomposition model to identify consistent human brain network modules. In *ICDM*, pages 301–310, 2016.
- [81] Dongsheng Luo, Wei Cheng, et al. Parameterized explainer for graph neural network. *arXiv preprint arXiv:2011.04573*, 2020.
- [82] Lei Luo, Jie Xu, Cheng Deng, and Heng Huang. Robust metric learning on grassmann manifolds with generalization guarantees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4480–4487, 2019.
- [83] Lei Luo, Yanfu Zhang, and Heng Huang. Adversarial nonnegative matrix factorization. In *International Conference on Machine Learning*, pages 6479–6488. PMLR, 2020.
- [84] Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.
- [85] Kenneth Marek et al. The parkinson progression marker initiative (ppmi). *Progress in neurobiology*, 95(4):629–635, 2011.
- [86] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.
- [87] Tomas Mikolov et al. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [88] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.
- [89] Volodymyr Mnih et al. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

- [90] Daniel Moyer, Greg Ver Steeg, Chantal MW Tax, and Paul M Thompson. Scanner invariant representations for diffusion mri harmonization. *Magnetic resonance in medicine*, 84(4):2174–2189, 2020.
- [91] Seth A Myers, Aneesh Sharma, Pankaj Gupta, and Jimmy Lin. Information network or social network? the structure of the twitter follow graph. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 493–498, 2014.
- [92] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.
- [93] Peter F Neher et al. A machine learning based approach to fiber tractography using classifier voting. In *MICAI*, pages 45–52, 2015.
- [94] Mathias Niepert et al. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023, 2016.
- [95] Alexander Norcliffe, Cristian Bodnar, et al. On second order behaviour in augmented neural odes. *arXiv preprint arXiv:2006.07220*, 2020.
- [96] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.
- [97] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- [98] Geoffrey JM Parker et al. A framework for a streamline-based probabilistic index of connectivity (pico) using a structural interpretation of mri diffusion measurements. *Journal of Magnetic Resonance Imaging: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 18(2):242–254, 2003.
- [99] Adam Paszke et al. Automatic differentiation in pytorch. 2017.
- [100] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

- [101] Marco Pennacchiotti and Siva Gurumurthy. Investigating topic models for social media user recommendation. In *Proceedings of the 20th international conference companion on World wide web*, pages 101–102, 2011.
- [102] Bryan Perozzi et al. Deepwalk: Online learning of social representations. In *ACM SIGKDD*, pages 701–710. ACM, 2014.
- [103] Michael Poli, Stefano Massaroli, et al. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.
- [104] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. Routledge, 2018.
- [105] Andres Potapczynski, Gabriel Loaiza-Ganem, and John P Cunningham. Invertible gaussian reparameterization: Revisiting the gumbel-softmax. *arXiv preprint arXiv:1912.09588*, 2019.
- [106] Philippe Poulin et al. Learn to track: Deep learning for tractography. In *MICCAI*, pages 540–547, 2017.
- [107] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1150–1160, 2020.
- [108] Scott C Ritchie et al. A scalable permutation approach reveals replication and preservation patterns of network modules in large datasets. *Cell systems*, 3(1):71–82, 2016.
- [109] Herbert Robbins and Sutton Monro. A stochastic approximation method. In *Herbert Robbins Selected Papers*, pages 102–109. Springer, 1985.
- [110] David E Rumelhart et al. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.
- [111] Deepak Saini, Arnav Kumar Jain, Kushal Dave, Jian Jiao, Amit Singh, Ruofei Zhang, and Manik Varma. Galaxc: Graph neural networks with labelwise attention for extreme classification. In *Proceedings of the Web Conference 2021*, pages 3733–3744, 2021.

- [112] Michael Schlichtkrull et al. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.
- [113] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.
- [114] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pages 488–495. PMLR, 2009.
- [115] David I Shuman et al. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.
- [116] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [117] Olaf Sporns et al. The human connectome: a structural description of the human brain. *PLoS computational biology*, 1(4):e42, 2005.
- [118] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019.
- [119] Xiangguo Sun, Hongzhi Yin, Bo Liu, Hongxu Chen, Qing Meng, Wang Han, and Jiuxin Cao. Multi-level hyperedge distillation for social linking prediction on sparsely observed networks. In *Proceedings of the Web Conference 2021*, pages 2934–2945, 2021.
- [120] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.
- [121] Christian Szegedy et al. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [122] Haoteng Tang, Lei Guo, et al. Hierarchical brain embedding using explainable graph learning. In *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, pages 1–5. IEEE, 2022.

- [123] Haoteng Tang, Guixiang Ma, et al. Commpool: An interpretable graph pooling framework for hierarchical graph representation learning. *Neural Networks*, 143:669–677, 2021.
- [124] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.
- [125] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, page 613–622, USA, 2006. IEEE Computer Society.
- [126] Nathalie Tzourio-Mazoyer et al. Automated anatomical labeling of activations in spm using a macroscopic anatomical parcellation of the mni mri single-subject brain. *Neuroimage*, 15(1):273–289, 2002.
- [127] Johan Ugander, Lars Backstrom, Cameron Marlow, and Jon Kleinberg. Structural diversity in social contagion. *Proceedings of the National Academy of Sciences*, 109(16):5962–5966, 2012.
- [128] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.
- [129] Ilya M Veer et al. Whole brain resting-state analysis reveals decreased functional connectivity in major depression. *Frontiers in systems neuroscience*, 4:41, 2010.
- [130] Petar Veličković, Guillem Cucurull, et al. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
- [131] Petar Velickovic et al. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 1(2), 2017.
- [132] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.
- [133] Daixin Wang et al. Structural deep network embedding. In *ACM SIGKDD*, pages 1225–1234. ACM, 2016.

- [134] De Wang et al. Human connectome module pattern detection using a new multi-graph minmax cut model. In *MICCAI*, pages 313–320, 2014.
- [135] Qi Wang, Lei Guo, Paul M Thompson, Clifford R Jack Jr, Hiroko Dodge, Liang Zhan, Jiayu Zhou, Alzheimer’s Disease Neuroimaging Initiative, et al. The added value of diffusion-weighted mri-derived structural connectome in evaluating mild cognitive impairment: A multi-cohort validation. *Journal of Alzheimer’s Disease*, 64(1):149–169, 2018.
- [136] Qi Wang, Mengying Sun, Liang Zhan, Paul Thompson, Shuiwang Ji, and Jiayu Zhou. Multi-modality disease modeling via collective deep matrix factorization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1155–1164, 2017.
- [137] Qi Wang, Liang Zhan, Paul M Thompson, Hiroko H Dodge, and Jiayu Zhou. Discriminative fusion of multiple brain networks for early mild cognitive impairment detection. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 568–572. IEEE, 2016.
- [138] Shen Wang et al. Structural deep brain network mining. In *ACM KDD*, pages 475–484. ACM, 2017.
- [139] Xinjun Wang, Zhe Sun, Yanfu Zhang, Zhongli Xu, Hongyi Xin, Heng Huang, Richard H Duerr, Kong Chen, Ying Ding, and Wei Chen. Brem-sc: a bayesian random effects mixture model for joint clustering single cell multi-omics data. *Nucleic acids research*, 48(11):5814–5824, 2020.
- [140] Xinjun Wang, Zhongli Xu, Haoran Hu, Xueping Zhou, Yanfu Zhang, Robert Lafyatis, Kong Chen, Heng Huang, Ying Ding, Richard H Duerr, et al. Secant: a biology-guided semi-supervised method for clustering, classification, and annotation of single-cell multi-omics. *PNAS nexus*, 1(4):pgac165, 2022.
- [141] Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. Dissecting the diffusion process in linear graph convolutional networks. *arXiv preprint arXiv:2102.10739*, 2021.
- [142] Felix Wu, Amauri Souza, et al. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.



- [143] Shu Wu, Yuyuan Tang, et al. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 346–353, 2019.
- [144] Wei Wu, Bin Li, Chuan Luo, and Wolfgang Nejdl. Hashing-accelerated graph neural networks for link prediction. In *Proceedings of the Web Conference 2021*, pages 2910–2920, 2021.
- [145] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018.
- [146] Louis-Pascal Xhonneux, Meng Qu, and Jian Tang. Continuous graph neural networks. In *International Conference on Machine Learning*, pages 10432–10441. PMLR, 2020.
- [147] Wenhan Xian, Feihu Huang, Yanfu Zhang, and Heng Huang. A faster decentralized algorithm for nonconvex minimax problems. *Advances in Neural Information Processing Systems*, 34:25865–25877, 2021.
- [148] Qianqian Xie, Jimin Huang, Pan Du, Min Peng, and Jian-Yun Nie. Graph topic neural network for document representation. In *Proceedings of the Web Conference 2021*, pages 3055–3065, 2021.
- [149] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.
- [150] Noriaki Yahata et al. A small number of abnormal brain connections predicts adult autism spectrum disorder. *Nature communications*, 7:11254, 2016.
- [151] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1365–1374, 2015.
- [152] Rex Ying, Dylan Bourgeois, et al. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32:9240, 2019.
- [153] Rex Ying et al. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

- [154] Zhitao Ying et al. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pages 4805–4815, 2018.
- [155] Kosuke Yoshida et al. Prediction of clinical depression scores and detection of changes in whole-brain using resting-state functional mri data with partial least squares regression. *PloS one*, 12(7):e0179638, 2017.
- [156] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. *arXiv preprint arXiv:2106.07594*, 2021.
- [157] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33, 2020.
- [158] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *Proceedings of the Web Conference 2021*, pages 413–424, 2021.
- [159] Hao Yuan, Jiliang Tang, et al. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 430–438, 2020.
- [160] Ling-Li Zeng et al. Identifying major depression using whole-brain functional connectivity: a multivariate pattern analysis. *Brain*, 135(5):1498–1507, 2012.
- [161] Liang Zhan et al. Comparison of nine tractography algorithms for detecting abnormal structural brain networks in alzheimer’s disease. *Frontiers in aging neuroscience*, 7:48, 2015.
- [162] Chi Zhang et al. Deepemd: Differentiable earth mover’s distance for few-shot learning. *arXiv e-prints*, pages arXiv–2003, 2020.
- [163] Fanjin Zhang, Xiao Liu, Jie Tang, Yuxiao Dong, Peiran Yao, Jie Zhang, Xiaotao Gu, Yan Wang, Bin Shao, Rui Li, et al. Oag: Toward linking large-scale heterogeneous entity graphs. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2585–2595, 2019.
- [164] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. Prone: Fast and scalable network representation learning. In *IJCAI*, volume 19, pages 4278–4284, 2019.

- [165] Jing Zhang et al. Panther: Fast top-k similarity search on large networks. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1445–1454, 2015.
- [166] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- [167] Xi Zhang et al. Multi-view graph convolutional network and its applications on neuroimage analysis for parkinson’s disease. *arXiv preprint arXiv:1805.08801*, 2018.
- [168] Xian Zhang et al. Can depression be diagnosed by response to mother’s face? a personalized attachment-based paradigm for diagnostic fmri. *PloS one*, 6(12):e27253, 2011.
- [169] Yanfu Zhang, Runxue Bao, Jian Pei, and Heng Huang. Toward unified data and algorithm fairness via adversarial data augmentation and adaptive model fine-tuning. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 1317–1322. IEEE, 2022.
- [170] Yanfu Zhang, Li Ding, and Gaurav Sharma. A local-linear-fitting-based matting approach for accurate depth upsampling. In *2016 IEEE Western New York Image and Signal Processing Workshop (WNYISPW)*, pages 1–5. IEEE, 2016.
- [171] Yanfu Zhang, Li Ding, and Gaurav Sharma. Hazerd: an outdoor scene dataset and benchmark for single image dehazing. In *2017 IEEE international conference on image processing (ICIP)*, pages 3205–3209. IEEE, 2017.
- [172] Yanfu Zhang, Li Ding, and Gaurav Sharma. Local-linear-fitting-based matting for joint hole filling and depth upsampling of rgb-d images. *Journal of Electronic Imaging*, 28(3):033019–033019, 2019.
- [173] Yanfu Zhang, Hongchang Gao, et al. Robust self-supervised structural graph neural network for social network prediction. In *Proceedings of the ACM Web Conference 2022*, pages 1352–1361, 2022.
- [174] Yanfu Zhang, Shangqian Gao, and Heng Huang. Exploration and estimation for model compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 487–496, 2021.

- [175] Yanfu Zhang, Shangqian Gao, and Heng Huang. Recover fair deep classification models via altering pre-trained structure. In *European Conference on Computer Vision*, pages 481–498. Springer, 2022.
- [176] Yanfu Zhang, Shangqian Gao, Jian Pei, and Heng Huang. Improving social network embedding via new second-order continuous graph neural networks. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 2515–2523, 2022.
- [177] Yanfu Zhang and Heng Huang. New graph-blind convolutional network for brain connectome data analysis. In *International Conference on Information Processing in Medical Imaging*, pages 669–681. Springer, 2019.
- [178] Yanfu Zhang, Lei Luo, and Heng Huang. Unified fairness from data to learning algorithm. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1499–1504. IEEE, 2021.
- [179] Yanfu Zhang, Lei Luo, Wenhan Xian, and Heng Huang. Learning better visual data similarities via new grouplet non-euclidean embedding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9918–9927, 2021.
- [180] Yanfu Zhang, Liang Zhan, Weidong Cai, Paul Thompson, and Heng Huang. Integrating heterogeneous brain networks for predicting brain disease conditions. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 214–222. Springer, 2019.
- [181] Yanfu Zhang, Liang Zhan, Paul M Thompson, and Heng Huang. Biological knowledge guided deep neural network for brain genotype-phenotype association study. In *Multimodal Brain Image Analysis and Mathematical Foundations of Computational Anatomy*, pages 84–92. Springer, 2019.
- [182] Yanfu Zhang, Liang Zhan, Shandong Wu, Paul Thompson, and Heng Huang. Disentangled and proportional representation learning for multi-view brain connectomes. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part VII 24*, pages 508–518. Springer, 2021.
- [183] Da Zheng, Minjie Wang, et al. Learning graph neural networks with deep graph library. In *Companion Proceedings of the Web Conference 2020*, pages 305–306, 2020.

- [184] Yu Zheng, Chen Gao, Liang Chen, Depeng Jin, and Yong Li. Dgcn: Diversified recommendation with graph convolutional networks. In *Proceedings of the Web Conference 2021*, pages 401–412, 2021.
- [185] Jason Zhu, Yanling Cui, Yuming Liu, Hao Sun, Xue Li, Markus Pelger, Tianqi Yang, Liangjie Zhang, Ruofei Zhang, and Huasha Zhao. Textgcn: Improving text encoder via graph neural network in sponsored search. In *Proceedings of the Web Conference 2021*, pages 2848–2857, 2021.
- [186] Jiong Zhu, Yujun Yan, et al. Generalizing graph neural networks beyond homophily. *arXiv preprint arXiv:2006.11468*, 2020.
- [187] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pages 2069–2080, 2021.
- [188] Juntang Zhuang, Nicha Dvornek, et al. Ordinary differential equations on graph networks. 2019.