Graph Convolutional Network with Novel Normalization Layers for Connectomes Analysis

Rui Shen
ruishen@seas.upenn.edu

Rongguang Wang
rgw@seas.upenn.edu

Chenyang Fang
cyfang@wharton.upenn.edu

Abstract

Brain connectomes is a comprehensive map that shows the structural and functional connectivity of neural pathways in the human brain. Analyzing the information encoded by connectomes can promote a critical understanding of the early diagnosis of many neuropsychiatric disorders, such as Autism Spectrum Disorder (ASD). Brain connectomes data is usually represented as a graph with a connectivity matrix illustrating the strength of neural connections (e.g. temporal correlation of brain activities) between brain regions. In contrast to classical graph analysis methods that depend on hand-engineering descriptors, recent progress of extending deep learning approaches to non-grid data have opened new opportunities for building predictive models for brain connectomes in a data-driven learning fashion. However, due to the existence of large noise and limited training samples, directly applying a deep network is prone to over-fitting. In this work, we present a novel graph convolutional neural network (GCN) with node-wise batch normalization and embedding normalization [1] for better generalization. We performed experiments on the ABIDE dataset and achieved state-of-the-art 68.7% classification accuracy.

1 Introduction

Functional connectivity, as often captured by correlations between resting-state functional MRI (rs-fMRI) signals, has produced novel insights linking differences in brain organization to the individual or group-level characteristics. Recently, machine learning models are being increasingly applied to study and exploit individual variation in functional connectivity data. Khosla et al. [1] demonstrated a volumetric Convolutional Neural Network (CNN) framework for Autism Spectrum Disorder (ASD) classification by concatenating voxel-level connectivity maps. On the other hand, node-level connectivity is more commonly used in network neuroscience to illustrate the underlying mechanisms of complex brain function [2]. The advances in graph analysis and the generalization of deep neural networks to non-grid data make node-level connectomes a more attractive tool to analyze brain relationships in a large population of subjects. Zhang et al. [3] used a graph convolutional network (GCN) to extract graph embedding from connectomes and exploit them in inferring the graph structure of the data.

Handling node-level functional connectivity poses a series of challenges. First, Given the sensitivity of rs-fMRI measurements to physiological variables, the acquired signals usually suffer from low test-retest reliability [4]. Noise and artifacts are further accumulated due to the limitation of current prepossessing methods for connectomes data. Second, connectomes, representing complex brain functional networks, are high-dimensional data with entries in the scale of 10^4 or even 10^6. However, most existing datasets only consist of a few hundred subjects. Due to those factors, neural network models face severe over-fitting problems when training on connectomes.

In this work, we proposed a GCN framework with two novel embedding normalization layers for connectomes-based classification. We tested our method on a well-known dataset, The Autism Brain

[1] code available at https://github.com/Sherry-SR/dl_graph_connectomes

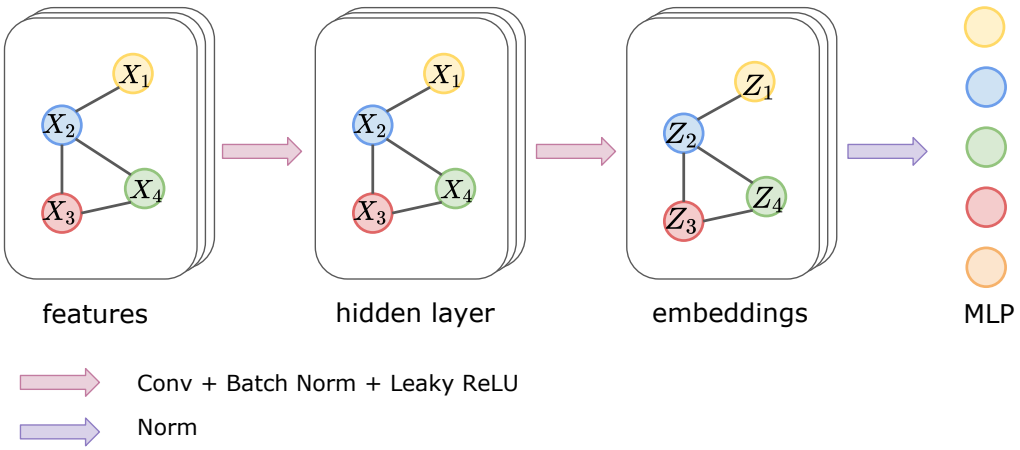

Figure 1: Proposed multi-layer Graph Convolutional Network (GCN).

Imaging Data Exchange (ABIDE) [5], and compare it with other conventional machine learning and deep learning methods. We showed that our method is easier to train and can achieve better generalization of the network.

2 Method

In this section, we described our approach for classifying subjects with ASD. Figure 1 illustrates an overview of the proposed approach for ASD classification. In the proposed method, we used the first-order approximation of GCN in the first stage to obtain embedding for a graph, and a multi-layer perceptron (MLP) in the second stage for the classification task. Multiple normalization tricks were used to improve the stability in the training process and achieve better generalization. A Dropout layer was used between MLP to prevent further over-fitting.

2.1 Graph covolutional layer

CNNs enable extraction of statistical features from grid-structured data, in the form of local stationary patterns, and their aggregation for different semantic analysis tasks, e.g. image recognition [6]. When the signal of interest does not lie on a regular domain, e.g. graphs, direct application of CNNs might be problematic due to the presence of convolution and pooling operators which typically defined on grids. Generalizing the convolution to graphs is a common way to address this problem.

A graph can be represented as $\{V, E, W\}$, with $V = \{v_1, v_2, \ldots, v_n\}$ the set of n vertices, $E \subseteq V \times V$ the set of m edges, and $W \in \mathcal{R}^{n \times n}$ the connectivity matrix of the graph. To perform graph convolution, one strategy is to conduct the operation in the frequency domain using graph Laplacian, which is feasible according to the convolution theorem. Graph Laplacian is typically defined as

$$L_n = I_n - D^{-1/2} W D^{-1/2}$$

in the normalized form, where $D \in \mathcal{R}^{n \times n}$ the degreee matrix with the diagonal entries $d_{ii} = \sum_j w_{i,j}$ and I_n a identity matrix. Laplacian matrix is positive semi-definite, such that the eigenvalue decomposition $L = U^T \Lambda U$ exists. $U = [u_0, u_1, \ldots, u_{n-1}]$ specifies Fourier basis and graph Fourier transform is defined as $\hat{x} = U^T x$, with $x \in R^n$ the signal. Graph convolution is then defined as

$$y = U g_\theta(\Lambda) U^T x$$

According to K-order Chebyshev polynomial [7], the filter g_θ can be approximated using

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\Lambda)$$

with θ_k the trainable parameters and $T_k(\Lambda)$ the polynomials. If we truncate the Chebyshev polynomial to only first order, graph convolutional layer can be reformulated as

$$y = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}x\Theta)$$

with $\tilde{A} = A + I_n$, $\tilde{D}$ similarly defined as D, Θ trainable parameters, and $\sigma(\cdot)$ an activation function.

2.2 Normalization for nodes

Two types of novel normalization layers were used to help the stability and convergence in the training process. Between each graph convolutional layer, a normalization layer was used to reduce the "covariate shift" for each node. Unlike traditional batch normalization which normalizes the summed inputs to each channel over the training cases [8], node normalization applies a similar operation but for each node in the graph. It can be formulated as

$$\tilde{x}^v_{i,j} = \frac{x^v_{i,j} - E_{i\in\mathcal{B},j\in\mathcal{C}}[x^v_{i,j}]}{\sqrt{Var_{i\in\mathcal{B},j\in\mathcal{C}}[x^v_{i,j}]}} \times \gamma^v + \beta^v$$

where $\mathcal{B}$ and $\mathcal{C}$ contains indices for each sample in the mini-batch and each channel of the filters respectively. The node normalization has similar advantages as batch normalization, such as enabling a higher learning rate and regularizing the model. Additionally, as the graph embedding is generated in the form of nodes, node normalization which balances the nodes of the graph can improve the stability and efficiency in extracting embedding.

2.3 Normalization for graph embedding

Another normalization layer was applied between GCN and MLP for each sample over the dimension of graph embedding, which is defined as

$$\tilde{x}^k_i = \frac{x^k_i - E_{k\in\mathcal{K}}[x^k_i]}{\sqrt{Var_{i\in\mathcal{K}}[x^k_i]}}$$

where $\mathcal{K}$ stands for the embedding dimensions. This normalization is actually equivalent to layer normalization [9] or instance normalization [10] when input size equals to 1. This layer can act like contrast normalization over the embedding. It can stabilize the hidden state dynamics and keep characteristics for an individual sample at the same time. No running average needs to be calculated during training, and large noise in one sample will not affect each other in the mini-batch.

3 Dataset & Pre-processing

3.1 The Autism Brain Imaging Data Exchange (ABIDE)

ABIDE is a multi-site dataset openly sharing anatomical and functional brain imaging data of 539 individuals diagnosed with Autism Spectrum Disorder (ASD), as well as 573 normal controls (NC). We used the data processed by the Configurable Pipeline for the Analysis of Connectomes (CPAC) [11]. This pipeline performs motion correction, global mean intensity normalization and standardization of fMRI data to MNI space ($3 \times 3 \times 3$ mm resolution). We use ABIDE data that passed quality control (QC) by all the functional raters. This yielded a final sample size of 774 subjects, comprising 379 ASD and 395 NC. In this project, we consider parcellation Dosenbach 160 (DOS 160, $N = 161$) [12].

3.2 Functional Connectomes

Functional connectomes were generated by first averaging rs-fMRI time series for each ROI, then calculating the temporal Pearson correlation coefficients between each pair of averaged fMRI signals. The resulting connectomes data can be represented using a $N \times N$ connectivity matrix illustrating the strength of neural connections during brain activities.

4 Experiments

In our experiments, we performed a node-level connectomes-based classification task of ASD v.s. NC using 4 deep learning frameworks: GCN, GCN with batch normalization (GCN-B), GCN with embedding normalization (GCN-E), and GCN with both node and embedding normalization (GCN-NE). We conducted 10-fold cross-validation on the ABIDE dataset. Results are also compared with other state-of-art voxel-level learning methods.

4.1 Training details

The proposed GCN-NE model was implemented using PyTorch [13]. We applied two GCN layers with hidden neurons of 40 to extract graph embedding, and two fully connected layers with hidden neurons of 50 to perform classification. Normalization for nodes was used between GCN layers. Graph embedding was vectorized from GCN output and normalized for each sample. A dropout layer with $p = 0.5$ was used in MLP. Adam with the initial learning rate of $1\mathrm{e}{-1}$ and weight decay of $1\mathrm{e}{-3}$ was used in training. The learning rate was reduced every 100 epochs with $\gamma = 0.1$. A batch size of 64 was used. The model was trained on a single Nvidia 1060Ti GPU. The results were reported on training $2,000$ iterations.

For comparison, similar hyper-parameters were used in GCN, GCN-B, and GCN-E frameworks, but without using any normalization in GCN, only with embedding normalization in GCN-E, and replaced node & embedding normalization with two batch normalization layers in GCN-B. For GCN and GCN-B, the learning rate was adjusted to $1\mathrm{e}^{-}3$ for better performance.

4.2 Results and discussion

In Table 1 we provide 10-fold cross-validation results obtained with GCN-B, GCN-E, and GCN-NE. Training for basic GCN with similar hyper-parameters either failed to achieve higher than 55% accuracy, which means only slighter better than a random guess. Adding normalization layers improved classification performance significant. Compared with batch normalization, the proposed normalization layers for node and embedding can achieve better accuracy and are more stable over different folds. It is worth noting that with proposed normalization layers, we can further enable a larger learning rate. Table 2 shows a comparison with previously reported methods using voxel-level connectomes. Our method outperformed others even with much fewer connectomes features. Meanwhile, node-level connectomes data is easier to generate and more meaningful in network neuroscience studies.

Table 1: Comparison with Node-level Learning Methods.

Fold	0	1	2	3	4	5	6	7	8	9	Average
GCN-B	61.0	58.4	66.2	59.7	63.6	72.7	62.3	63.6	66.2	64.9	63.9
GCN-E	**68.6**	62.3	66.2	68.8	64.9	62.3	**72.7**	66.2	68.8	**66.2**	66.7
GCN-NE	66.2	**67.5**	**67.5**	**70.1**	**67.5**	**75.3**	64.9	**71.4**	**70.2**	**66.2**	**68.7**

* GCN failed to converge or achieved lower than 55% accuracy.

Table 2: Comparison with State-of-art Voxel-level Learning Methods [1]

Methods	Ridge	SVM (l_1)	SVM (l_2)	FCN	3D-CNN	GCN-NE (ours)
Accuracy	66.7	65.3	66.7	67.2	68.6	**68.7**

5 Conclusion

In this paper, we proposed a GCN framework with novel normalization layers for node-level connectomes-based classification. We demonstrated that the proposed node and graph embedding normalization layers can achieve better generalization and stability than traditional batch normalization. We also showed that our method is a more attractive solution in network neuroscience study, with comparable or even better classification accuracy than many other learning methods for voxel-level connectomes.

References

[1] M. Khosla, K. Jamison, A. Kuceyeski, and M. R. Sabuncu, "3d convolutional neural networks for classification of functional connectomes," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pp. 137–145, Springer, 2018.

[2] D. S. Bassett and O. Sporns, "Network neuroscience," *Nature neuroscience*, vol. 20, no. 3, p. 353, 2017.

[3] Y. Zhang and H. Huang, "New graph-blind convolutional network for brain connectome data analysis," in *International Conference on Information Processing in Medical Imaging*, pp. 669–681, Springer, 2019.

[4] X.-N. Zuo, B. B. Biswal, and R. A. Poldrack, "Reliability and reproducibility in functional connectomics," *Frontiers in neuroscience*, vol. 13, p. 117, 2019.

[5] C. Craddock, Y. Benhajali, C. Chu, F. Chouinard, A. Evans, A. Jakab, B. S. Khundrakpam, J. D. Lewis, Q. Li, M. Milham, *et al.*, "The neuro bureau preprocessing initiative: open sharing of preprocessed neuroimaging data and derivatives," *Frontiers in Neuroinformatics*, vol. 7, 2013.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[9] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[10] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[11] C. Craddock, S. Sikka, B. Cheung, R. Khanuja, S. S. Ghosh, C. Yan, Q. Li, D. Lurie, J. Vogelstein, R. Burns, *et al.*, "Towards automated analysis of connectomes: The configurable pipeline for the analysis of connectomes (c-pac)," *Front Neuroinform*, vol. 42, 2013.

[12] N. U. Dosenbach, B. Nardos, A. L. Cohen, D. A. Fair, J. D. Power, J. A. Church, S. M. Nelson, G. S. Wig, A. C. Vogel, C. N. Lessov-Schlaggar, *et al.*, "Prediction of individual brain maturity using fmri," *Science*, vol. 329, no. 5997, pp. 1358–1361, 2010.

[13] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.