Comparing Shallow and Deep Graph Models for Brain Network Analysis

Erica Choi*
Department of Mathematics
Columbia University
New York, NY
eyc2130@columbia.edu

Sally Smith*
Department of Mathematics
Georgia Institute of Technology
Atlanta, GA
sallysmith@gatech.edu

Ethan Young*
Department of Mathematics
University of California, Los Angeles
Los Angeles, CA
young.j.ethan@gmail.com

***Abstract*—It is well-established that graph neural networks (GNNs) can effectively model networked data in a variety of fields. However, whether GNNs can outperform traditional shallow graph classification models such as graph kernels for brain network analysis remains unclear. To this end, we analyze different approaches for modeling brain networks, including graph kernel based SVM, basic GNNs and kernelized GNNs. These models are designed to aid in the analysis of diseases and mental disorders such as bipolar disorder, human immunodeficiency virus (HIV), post-traumatic stress disorder (PTSD), and depression. In particular, we conduct experiments with three methods: kernelized support vector machines (SVM), message passing graph neural networks (MPGNNs), and kernel graph neural networks (KerGNN). We conclude that 1) deep models (GNNs) generally outperform shallow models (SVM) and 2) models considering specific graph motifs do not seem to significantly improve performance. We also identify other graph kernels and GNN frameworks that show promise in motivating further research in brain network analysis.**

***Index Terms*—brain networks, GNNs, graph learning, graph kernels, neuroimaging data, SVM**

I. Introduction

The human brain is a complex organ that organizes and dictates all of the functions within the body. Neuroscience research has expanded in recent decades, resulting in rapid progress in studying the structures and functions of the human brain. Recent studies in neuroscience and brain imaging have reached the consensus that the interactions among brain regions are driving factors for neural development and disorders, but determining what types of mathematical models should be used to analyze such interactions remains an active area of research.

One of the biggest limitations to progress in neuroscience research is the invasive nature of many existing methods for studying the brain. Comparative studies have found a significant relationship between invasive research methods such as histological research and noninvasive methods such as structural neuroimaging [1]. These results motivate refinement and innovation in noninvasive research methods for neuroscience. The field of network neuroscience has only recently been established [1], and research in this area seeks to leverage the power of modern computational methods and brain network modeling in conjunction with neurobiological data collection methods in order to produce novel results within the greater field of neuroscience.

In this project, we explore different approaches for modeling brain networks. We analyze several methods: traditional shallow graph models, modern deep graph neural networks (GNNs), and GNNs incorporating graph kernels. The goal of these models, and of our research as a whole, is to aid in the analysis of diseases and mental disorders such as bipolar disorder, HIV, PTSD, depression, and substance misuse and to harness modern computational methods to improve classification accuracy of pre-existing models that predict whether a brain is diseased or healthy.

Summary of Contributions: We adapt shallow and deep graph mining techniques and conduct a variety of experiments to benchmark their performance on brain network datasets. We also suggest several methods that show promise when applied to brain networks. Importantly, we hope that some of the ideas in these methods motivate further studies.

II. Preliminaries

A. Problem Formulation

The standard graph classification task considers the problem of classifying graphs into two or more categories. In this project, we perform binary classification on neuroimaging data to classify patients as either diseased or healthy. Our datasets consist of brain networks represented as weighted, undirected adjacency matrices constructed from fMRI scans. For more details on network construction, see Section 3 of [2]. Depending on the classification model, we further preprocess the datasets with methods such as threshold rounding.

B. Brain Networks

The brain network datasets $\mathcal{D} = \{\mathcal{G}_i, y_i\}_{i=1}^N$ consist of N subjects, where $\mathcal{G}_i = \{\mathcal{V}_i, \mathcal{E}_i\}$ represents the brain network of subject i and $y_i \in Y$ is its corresponding neural disease label. In $\mathcal{D}$, the brain network $\mathcal{G}_i$ of every subject i involves the same set of M nodes (ROIs), i.e., $\forall i, \mathcal{V}_i = \{v_p\}_{p=1}^M$. Thus, the difference across subjects only lies in the edges $\mathcal{E}_i$ (connections among brain regions), which are often represented by

*Equal contribution. This work was done during the Computational Mathematics for Data Science REU/RET at Emory University in Summer 2022. All three authors were undergraduate students mentored by Dr. Carl Yang (j.carlyang@emory.edu).

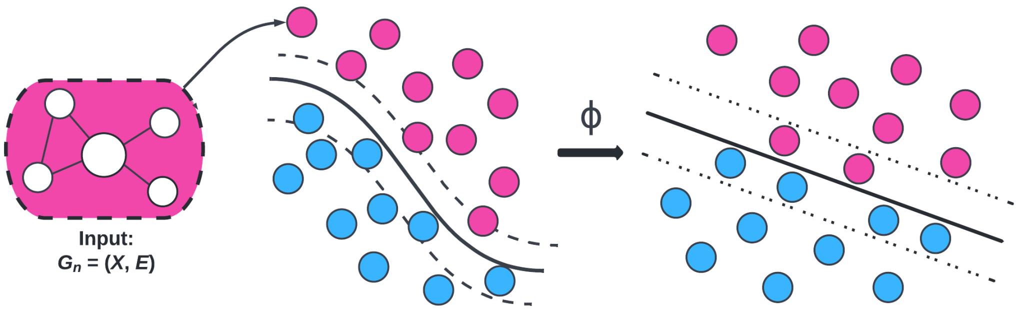


Fig. 1. Overview of graph kernelized learning.

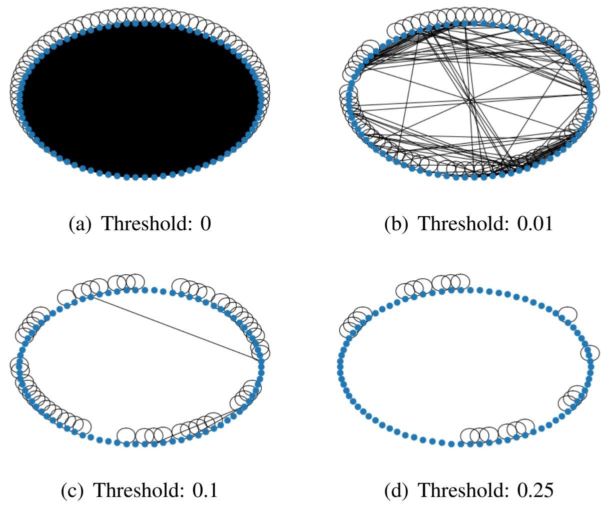


Fig. 2. Effect of threshold rounding on network density.

a weighted adjacency matrix $\mathbf{W}_i \in \mathbb{R}^{M\times M}$ per subject i. The edge weights in $\mathbf{W}$ are real-valued and can be both positive and negative.

C. Datasets

We are working with two datasets, one classifying human immunodeficiency virus (HIV) and one classifying bipolar disorder (BP). Each dataset consists of functional magnetic resonance imaging (fMRI) scans and classification labels in the form of integers, where 1 indicates a healthy patient and -1 indicates an unhealthy patient. Both datasets have been processed for us. More details can be found in Section 3 of [3], which goes more in depth about preprocessing fMRI data and constructing the brain networks using a set of regions of interest (ROIs) that define the network nodes.

D. Threshold Rounding

The brain networks are extremely dense and noisy. We address these issues in several of our experiments by implementing threshold rounding. That is, if the ij-th entry of adjacency matrix $\mathbf{W}$ is greater than or equal to the threshold, the entry is rounded to 1. Otherwise, the entry is rounded to 0. This additional step in data preprocessing allows us to sparsify our brain networks while preserving as much relevant structural information as possible. Fig. 2 shows how the density of a random observation decreases as our threshold increases. Note that even for ”small” thresholds such as 0.1, the brain network becomes extremely sparse and mostly consists of self-loops.

III. Shallow Models

Our first classification method is a ”shallow” model: kernelized **support vector machines** (SVM) [4]. This is a popular approach to graph classification that uses graph kernels to compute a kernel matrix K of size $n\times n$, where K_{ij} represents the similarity between G_i and G_j [5], and then plugging the computed kernel matrix into a kernelized learning algorithm. Graph kernels are appealing to implement because they can be computed as an inner product between pairs of observations. These methods are considered shallow models because they do not consist of many layers of computations, unlike their deep model counterparts. Fig. 1 visualizes this method.

A. Graph Kernels

We consider four graph kernels in our experiments: Weisfeiler-Lehman subtree (WL), Weisfeiler-Lehman optimal assignment (WLOA), shortest path (SP), and graphlet sampling (GS) kernels. For more details on graph kernels, please refer to [6] and Section 2 of [5].

1) Weisfeiler-Lehman Subtree Kernel: The WL kernel is a popular state-of-the-art algorithm employed in graph classification tasks. This kernel is built on the Weisfeiler-Lehman graph isomorphism test [7] and is essentially a relabeling procedure, where a node's labels are iteratively updated using the information of the node's neighbors. This kernel is computationally inexpensive, taking $O(hm)$ time, where h is the number of iterations and m is the number of edges.

2) Weisfeiler-Lehman Optimal Assignment Kernel: The next kernel we consider is the WLOA kernel, which leverages valid assignment kernels to improve the performance of the WL subtree kernel [8]. Similar to the WL kernel, the WLOA kernel is an iterative relabeling procedure that also considers node neighborhoods. This kernel is appealing in that it can be computed in linear time, taking $O(|X| + |Y|)$ time, where X and Y are elements of $[\mathcal{X}]^n$. $[\mathcal{X}]^n$ denotes the set of all n-element subsets of the set $\mathcal{X}$.

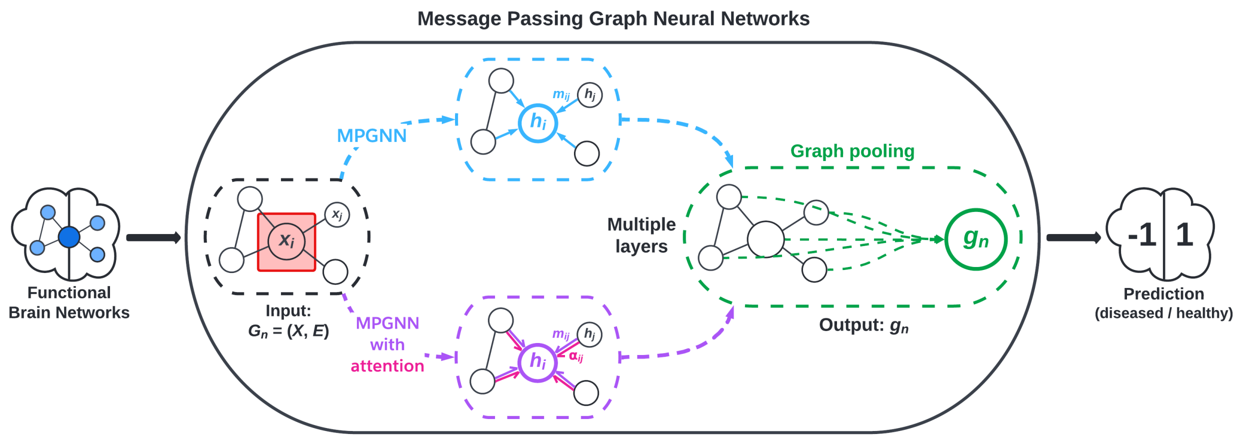


Fig. 3. BrainGB framework. Adapted from Fig. 1 of [2]. h_i is the node representation of node x_i, m_{ij} is the message from node x_j to x_i, and a_{ij} is the attention weight from node x_j to x_i.

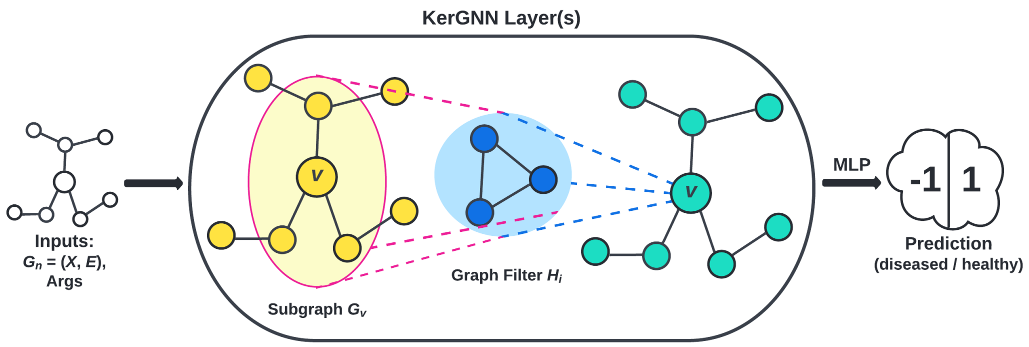


Fig. 4. KerGNN framework. Adapted from Fig. 3 in [9].

3) Shortest Path Kernel: The SP kernel is intuitive: it decomposes graphs into shortest paths, considers pairs of shortest paths, and compares their length and labels of path endpoints [10]. This kernel is computationally expensive, taking $O(n^4)$ time, where n is the number of nodes.

4) Graphlet Sampling Kernel: The GS kernel is also intuitive: it decomposes graphs into graphlets, which are subgraphs of k nodes, and compares the number of matching graphlets between two graphs [11]. This kernel is computationally intractable due to enumeration of all size-k subgraphs with large k, taking $O(n^k)$ time.

IV. Deep Models

Our second classification method is a "deep" model: **graph neural networks**. GNN architectures are an extremely vibrant area of research due to the diversity and abundance of graph data. For our experiments, we focus on BrainGB [2] and kernel graph neural networks [9].

A. Message Passing Graph Neural Networks

There exist many benchmarks for the classification of brain networks. For example, the BrainNNExplainer framework [12] is an interpretable model that performs well on the brain network classification task. The first GNN we perform classification with is **BrainGB** [2], another GNN framework that is geared toward brain network analysis.

BrainGB is a message passing GNN (MPGNN) that we implement using the open-source BrainGB Python package, which is built on the *Pytorch* and *Pytorch Geometric* libraries. In this section, we briefly explain the two main MPGNNs employed in BrainGB: graph convolutional networks (GCNs) and graph attention networks (GATs). Section 4 of [2] gives further details and motivation on the MPGNN design choices. Fig. 3 visualizes the MPGNN architecture.

1) Graph Convolutional Networks: Message passing schemes in GCNs are generally composed of two steps: message passing and update. The message passing step consists of nodes receiving messages from their neighbors via a message function. These messages are then aggregated. The update step consists of updating each node's embeddings based on the aggregated messages via a differentiable update function.

2) Graph Attention Networks: The attention mechanism in GATs is used to improve the message passing scheme and can provide interpretability over edge importance [2]. In this framework, node representations are updated with additional attention weights.

TABLE I
HYPERPARAMETER SEARCH RANGE

Number of epochs	100; 150; 200; 250; 300; 350; 400; 450; 500
Learning rate	10^{-2}; 10^{-3}; 10^{-4}; 10^{-5}; 10^{-6}
Dropout rate	0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9
Nodes in graph filter	2; 4; 6; 8; 10; 12; 14; 16; 18; 20
Subgraph size	5; 10; 15; 20
k-hop neighborhood	1; 2; 3
Max step of RW	1; 2; 3; 4; 5

TABLE II
PERFORMANCE REPORT[a]

Data	Method	Accuracy	F1	AUC
HIV	WL-0.21	$0.67_{\pm 0.17}$	—	—
	WLOA-0.21	$0.65_{\pm 0.17}$	—	—
	SP-0.01	$0.66_{\pm 0.20}$	—	—
	GS-0.03	$0.66_{\pm 0.18}$	—	—
	GCN-concat	$0.64_{\pm 0.15}$	$0.59_{\pm 0.20}$	$0.77_{\pm 0.20}$
	GAT-concat	$0.73_{\pm 0.16}$	$0.71_{\pm 0.17}$	$0.81_{\pm 0.19}$
	GCN-edge concat	$0.71_{\pm 0.11}$	$0.69_{\pm 0.12}$	$0.77_{\pm 0.17}$
	GAT-edge concat	$0.69_{\pm 0.18}$	$0.67_{\pm 0.19}$	$0.73_{\pm 0.24}$
	KerGNN	$0.64_{\pm 0.19}$	—	—
BP	WL-0.4	$0.63_{\pm 0.19}$	—	—
	WLOA-0.42	$0.66_{\pm 0.12}$	—	—
	SP-0.02	$0.64_{\pm 0.12}$	—	—
	GS-0.04	$0.62_{\pm 0.15}$	—	—
	GCN-concat	$0.53_{\pm 0.13}$	$0.51_{\pm 0.14}$	$0.54_{\pm 0.16}$
	GAT-concat	$0.53_{\pm 0.13}$	$0.50_{\pm 0.13}$	$0.57_{\pm 0.19}$
	GCN-edge concat	$0.63_{\pm 0.12}$	$0.61_{\pm 0.13}$	$0.61_{\pm 0.17}$
	GAT-edge concat	$0.52_{\pm 0.17}$	$0.51_{\pm 0.16}$	$0.59_{\pm 0.19}$
	KerGNN	$0.68_{\pm 0.16}$	—	—

[a]The highest score for each dataset is highlighted. For clarity, we also highlight the method that produced that score. In the case of ties, we choose the score with the lower standard deviation.

B. *Kernel Graph Neural Networks*

For the second GNN framework, we seek to combine the approaches of shallow and deep graph modeling. To this end, we study GNN architectures that integrate graph kernels and benchmark their performance on our datasets. These GNNs take advantage of both the higher-order structural information given by graph kernels and the local information given by MPGNNs. The method that is of particular interest to is the **kernel graph neural network** (KerGNN) [9].

The KerGNN framework is essentially an MPGNN where graph kernels are integrated into the message passing process. KerGNNs employ trainable hidden graphs as graph filters, which are analogous to the convolution filter in convolutional neural networks (CNNs). These filters are combined with subgraphs and use graph kernels to update node embeddings. Fig. 4 is a simplified overview of the KerGNN framework.

V. EXPERIMENTS

A. *Experiment Settings*

1) SVM: We compute the graph kernels using the Python package GraKel [13]. To account for the small number of observations in our datasets, we averaged classification accuracy over 25 different train-test splits. For threshold rounding, we picked an optimal threshold (one that maximized mean classification accuracy) by comparing the performance of thresholds ranging from 0 to 1 and incrementing by 0.01. For the WL and WLOA kernels, we used the default number of iterations, which is 5.

To get a handle on the overall performance of the WL and WLOA graph kernels, we also ran tests by varying the number of iterations. Specifically, we used the optimal threshold derived in the previous experiment and incremented from 1 to 20 by 1. For the GS kernel, we ran experiments with size-5 subgraphs as a compromise between sufficient information given by graphlet size and computational resources.

2) BrainGB: We conduct our experiments based on the settings outlined in Section V of [2]. In particular, we experimented with different combinations of node features, message passing schemes, and pooling strategies. To streamline our workflow, we focus on two message passing schemes in GCNs and GATs that performed well in [2]: concat and edge concat. Accuracy, F1 Score, and Area Under the ROC Curve (AUC) are the metrics evaluating performance. We use the AUC to highlight the best performing model because it is not sensitive to changes in the class distribution. We once again average performance over different train-test splits.

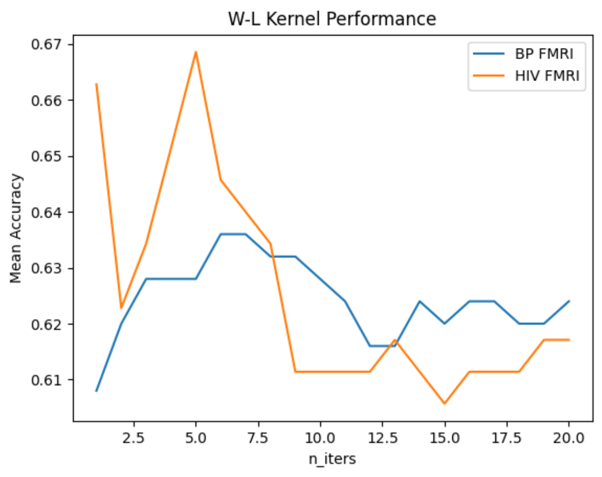
3) KerGNN: We conduct our experiments using the *Pytorch* implementation of KerGNN. We use a 0.05 rounding threshold and averaging performance over 20 train-val-test splits. The training set consists of 80% of the data, the validation set consists of 10%, and the testing set consists of the remaining 10%. The choice of a 0.05 rounding threshold is motivated by sparsifying the networks while preserving as many important connections as possible. However, we believe that experimenting with other thresholds and gaining insights into any impact threshold rounding has on brain network analysis to be a valuable endeavor.

Our experiments consider a single KerGNN layer with the random walk (RW) kernel. The choice of kernel follows from the extensive studies detailed in [9]. Due to time and computing resource constraints, we are unable to consider other worthwhile experiments, such as architectures with more than one KerGNN layer, different rounding thresholds, and other graph kernels.

We use grid search to select the set of hyperparameters that performs the best on our datasets. We consider hyperparameters related to neural network training, such as the number of epochs, learning rate, and dropout rate. We also consider those unique to KerGNN, such as the number of nodes in the graph filter, size of the subgraph, k-hop neighborhood, and maximum step for the RW. Due to computing resource constraints, we leave all other KerGNN hyperparameters to be default settings. Table I outlines our hyperparameter search range. For more details on the KerGNN framework, see [9].

B. *Results and Analysis*

Table II summarizes the classification performance of our methods on the HIV and BP datasets.



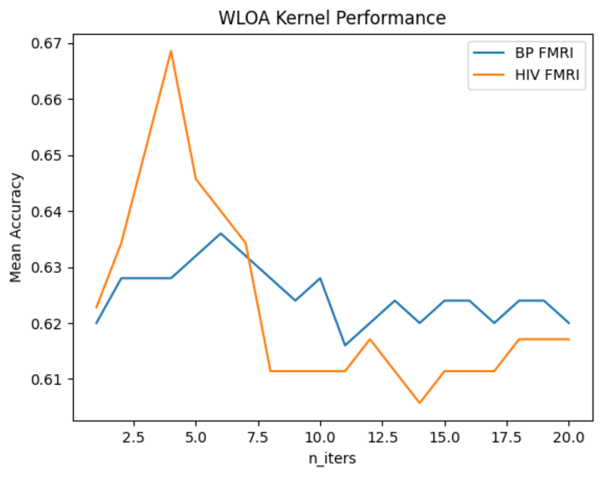


Fig. 5. Number of iterations (n_iters) experiments.

1) SVM: We see that for an optimal rounding threshold, both datasets had similar mean classification accuracy. The performance of our experiments had high standard deviation, which is reflective of the small size and inconsistent performance of our testing sets (poor splits of our datasets generally lead to low classification accuracy). For experiments varying the number of iterations (WL and WLOA kernels), Fig. 5 shows the mean classification accuracy seemingly converging as the number of iterations increased.

From the generally poor performance, we are uncertain whether there is higher-order information in brain networks useful for classification with graph kernels. For example, the WL and WLOA kernels iteratively update node labels based on the node's local information (e.g., subtrees in the WL kernel), but it seems that a node's neighborhood is unimportant in distinguishing between healthy and diseased patients' brain networks. For the SP and GS kernels that exploit network structure, their similar performance with the WL and WLOA kernels suggest that subgraph structures are not particularly helpful in our classification task.

2) BrainGB: Despite the small sizes of our datasets, it is clear that MPGNNs are effective on the HIV data; however, their poor performance on the BP data indicates their struggle with generalizing to other datasets. Like with the SVM experiments, we observe high standard deviations due to poor split assignments. Furthermore, while MPGNNs capture local node information well, they are prone to overfitting, especially on small datasets. They also lack some interpretability in their predictions. For a more thorough analysis on the performance of the BrainGB framework, please refer to [2].

3) KerGNN: KerGNN achieves the best performance on the BP dataset, but still suffers from high variance. Interestingly, the classification accuracy of the HIV dataset, after hyperparameter optimization, is worse. These results, while comparable to kernelized SVM, are in stark contrast with our BrainGB experiments that show consistently better performance on the HIV dataset than the BP dataset. Like in all of our previous experiments, the performance of KerGNN still exhibits high variance on both datasets due to some splits of our data performing extremely poorly.

VI. Conclusion

A. Discussion and Limitations

Our datasets are prohibitively small, which makes it difficult for robust model training. Consequently, poor split assignments occur frequently while running experiments, which is indicated in our results by extremely high standard deviations. We attempt to partially address this issue by averaging performance over many different splits; however, limits on time and computing resources make it difficult to run more comprehensive experiments. One additional experiment, for example, we want to consider is testing different rounding thresholds in the KerGNN experiments. Notably, across most experiments, the BP data shows consistently worse average performance than the HIV data.

There may be some explanations from neuroscience to address these problems. [12] observes that HIV significantly affects the connectivity within both the visual network (VN) and default mode network (DMN), while bipolar disorder mainly affects the bilateral limbic network (BLN). Because HIV patients' data show less connectivity across two sub-networks and BP patients' data show less connectivity in only one sub-network, it is harder to classify BP patients. [14] found that using multimodal neuroimagery increased their SVM model's performance in classifying bipolar disorder, so only utilizing a single modality (fMRI), rather than both fMRI data and MRI data, may also limit our model's classification accuracy on the BP dataset. [3] provides more details on the translation from brain imaging data to different sub-networks.

B. Future Work

Due to time constraints and implementation challenges, we were unable to conduct experiments with other graph mining techniques, which we briefly describe in this section. GNNs designed specifically for brain network analysis, such as [2], [12], and [15], generally involve some kind of message passing mechanism. Our interest in several of the following methods lies in their consideration of either graph kernels or graph structures (i.e., motifs), which makes them more expressive. We hope that these methods are effective and motivate further work in brain network analysis.

1) Graph Kernel Neural Networks: The first method of interest is **graph kernel neural networks** (GKNNs) [16], a GNN framework similar to KerGNN. This GNN architecture features the graph kernel convolution (GKC) layer, which is inspired by the convolution operator in CNNs. The flexibility of the GKC layer in accepting any number and type of graph kernels makes it particularly interesting in the context of our studies of graph kernels. For further details on the GKNN architecture, see [16]. The *Pytorch* implementation of GKNN is available online (the code may be downloaded as a ZIP file in the "Supplementary Material" section).

2) Graph Stochastic Attention: Next, we consider **graph stochastic attention** (GSAT) [17], an architecture that has the advantage of interpretability in its selection of relevant subgraphs, which, under certain assumptions outlined in [17], are optimal solutions of the graph information bottleneck objective. Please refer to [17] for more details. Applying this method to the MPGNN framework of BrainGB and comparing its performance on brain networks to that of GATs will hopefully yield insightful results. The *Pytorch* implementation of GSAT is available online.

3) k-dimensional Graph Neural Networks: We are also interested in **k-dimensional GNNs** (k-GNNs) [18], which are generalizations of GNNs based on the k-WL kernel that take into account higher-order graph structures. Importantly, [18] shows that k-GNNs effectively distinguish graph properties such as non-isomorphic subgraphs. This method is of theoretical interest because it explores the expressivity of GNNs. We hope that applying this method gives insights into which subgraph structures (if any) are relevant to brain network analysis. Please refer to [18] for more details. The *Pytorch Geometric* implementation of k-GNN is available online.

4) Message Passing Graph Kernels: The fourth method we consider is the **message passing graph kernel** (MPGK) [19]. The MPGK framework consists of iteratively and implicitly updating node representations, which then help construct the kernel that compares pairs of graphs. It may be interesting to compare the performance of MPGKs with the kernels in this study on brain networks. More details, such as relating the MPGK to the WL kernel, may be found in [19]. Code for MPGK is available online to compute kernel matrices.

5) Motif Convolutional Networks: Lastly, we are interested in **motif convolutional networks** (MCNs) [20], a model that generalizes GCNs by using motif adjacency matrices and finding relevant higher-order neighborhoods in graph data. A powerful feature of MCNs is that an additional attention mechanism is introduced to allow each node to select the most relevant motif. Importantly, we hope this method will give insights to the significance of graph motifs in brain network analysis. Please refer to [20] for more details on the MCN architecture.

Acknowledgment

We thank Dr. Lars Ruthotto, Dr. Carl Yang, Ms. Hejie Cui, Mr. Wei Dai and all other mentors of the Emory REU/RET Computational Mathematics for Data Science for their mentorship and technical assistance. This work was financially supported in part by the US National Science Foundation awards DMS-2051019 and DMS-1751636; this work was also partially supported by the internal funds and GPU servers provided by the Department of Computer Science of Emory University.

References

[1] D. S. B. . O. Sporns, "Network neuroscience," *Nature Neuroscience*, vol. 20, 2017.

[2] H. Cui, W. Dai, Y. Zhu, X. Kan, A. A. C. Gu, J. Lukemire, L. Zhan, L. He, Y. Guo, and C. Yang, "Braingb: A benchmark for brain network analysis with graph neural networks," 2022. [Online]. Available: https://arxiv.org/abs/2204.07054

[3] N. Chen, J. Shi, Y. Li, S. Ji, Y. Zou, L. Yang, Z. Yao, and B. Hu, "Decreased dynamism of overlapping brain sub-networks in major depressive disorder," *Journal of Psychiatric Research*, vol. 133, pp. 197–204, 2021.

[4] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The annals of statistics*, vol. 36, no. 3, pp. 1171–1220, 2008.

[5] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1365–1374.

[6] N. M. Kriege, F. D. Johansson, and C. Morris, "A survey on graph kernels," *Applied Network Science*, vol. 5, no. 1, 2020.

[7] B. Weisfeiler and A. Lehman, "The reduction of a graph to canonical form and the algebra which appears therein," *NTI*, vol. 9, pp. 12–16, 1968.

[8] N. Kriege, P.-L. Giscard, and R. Wilson, "On valid optimal assignment kernels and applications to graph classification," in *NeurIPS*, 2016, pp. 1615–1623.

[9] A. Feng, C. You, S. Wang, and L. Tassiulas, "Kergnns: Interpretable graph neural networks with graph kernels," in *AAAI*, 2022.

[10] K. M. Borgwardt and H.-P. Kriegel, "Shortest-path kernels on graphs," in *Fifth IEEE International Conference on Data Mining*, 2005, pp. 74–81.

[11] N. Przulj, "Biological network comparison using graphlet degree distribution," *Bioinformatics*, vol. 23, no. 2, pp. 177–183, 2007.

[12] H. Cui, W. Dai, Y. Zhu, X. Li, L. He, and C. Yang, "Brainnnexplainer: an interpretable graph neural network framework for brain network based disease analysis," in *ICML-IMLH*, 2021.

[13] G. Siglidis, G. Nikolentzos, S. Limnios, C. Giatsidis, K. Skianis, and M. Vazirgiannis, "Grakel: A graph kernel library in python," 2018. [Online]. Available: https://arxiv.org/abs/1806.02193

[14] H. Li, L. Cui, L. Cao, Y. Zhang, Y. Liu, W. Deng, and W. Zhou, "Identification of bipolar disorder using a combination of multimodality magnetic resonance imaging and machine learning techniques," in *BMC Psychiatry*, 2020.

[15] H. Cui, W. Dai, Y. Zhu, X. Li, L. He, and C. Yang, "Interpretable graph neural networks for connectome-based brain disorder analysis," 2022. [Online]. Available: https://arxiv.org/abs/2207.00813

[16] L. Cosmo, G. Minello, M. Bronstein, E. Rodolà, L. Rossi, and A. Torsello, "Graph kernel neural networks," 2021. [Online]. Available: https://arxiv.org/abs/2112.07436

[17] S. Miao, M. Liu, and P. Li, "Interpretable and generalizable graph learning via stochastic attention mechanism," 2022. [Online]. Available: https://arxiv.org/abs/2201.12987

[18] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," 2018. [Online]. Available: https://arxiv.org/abs/1810.02244

[19] G. Nikolentzos and M. Vazirgiannis, "Message passing graph kernels," 2018. [Online]. Available: https://arxiv.org/abs/1808.02510

[20] J. B. Lee, R. A. Rossi, X. Kong, S. Kim, E. Koh, and A. Rao, "Higher-order graph convolutional networks," 2018. [Online]. Available: https://arxiv.org/abs/1809.07697