2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) | 978-1-6654-9747-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/IPDPSW55747.2022.00043

Graph Convolutional Neural Networks for Alzheimer's Classification with Transfer Learning and HPC Methods

Anoop Kumar*, Vibha Balaji+, Chandrashekar M.A.+, Ambedkar Dukkipati*, Sathish Vadhiyar+
*Department of Computer Science and Automation
+Supercomputer Education and Research Centre
Indian Institute of Science, Bangalore, India
anoopkumar@iisc.ac.in, vibhabalaji1998@gmail.com, chandrashekr@iisc.ac.in, ambedkar@iisc.ac.in, vss@iisc.ac.in

***Abstract*—GCNs have been increasingly used for the classification of brain functional networks to aid early prediction of neurodegenerative diseases. It is important to analyze the performance and capabilities of these GCNs for the classification and obtain insights on how and when the GCNs can be used. In this work, we perform detailed analyses of the performance of GCNs for the classification of brain functional networks of Alzheimer's data. We study the impact of the various parameters including the thresholds used for graph generation, graph sizes, data sizes or the number of subjects, and classification accuracy across different visits of the subjects. We have also developed a transfer learning approach to train using one dataset and apply the weights on another dataset to make use of larger data sizes. Finally, we have developed GPU-based acceleration methods to decrease the training time. We find that the accuracy of the models improves when taking into account all visits of the subjects and also improves with increasing graph sizes. The use of transfer learning has also improved classification accuracy. Finally, the use of CUDA streams for asynchronous computations has resulted in a reduction in execution times by up to 60%.**

***Index Terms*—Brain functional networks, Alzheimer's data, GCNs, Transfer learning, GPU Acceleration.**

I. Introduction

Studying functional connectivity in human brain to aid early prediction of neurodegenerative diseases has been at the forefront of research [1] [2]. Much of the understanding of the human brain rests on the way how it is measured and modeled [3]. Exploring the human brain from the viewpoint of connectivity patterns reveals important information regarding the structural, functional, and causal organization of the brain [4]. With the launch of the Human Connectome Project [5], computational methods to study connectivity in brain have become prevalent. Brain functions can be learnt through neuroimaging techniques that assess changes in metabolism via positron emission tomography (PET) or changes in blood oxygenation level-dependent (BOLD) responses via functional MRI (fMRI). fMRI and PET offer low temporal resolution but significantly high spatial resolution. Among the two, PET is significantly costlier and involves usage of radioactive isotopes. Hence, fMRI data, especially rs-fMRI (resting state fMRI) data, has been widely used for brain studies [4]. Resting state functional MRI (rs-fMRI) provides us with information about the default state of the brain, and allows us to evaluate functional connectivity and its alterations in brain disorders. Brain functional connectivity (FC) derived from rs-fMRI data has become a powerful approach to measure and map brain activity.

In early rsfMRI studies, functional connectivity was often summarised by a few spatial maps spanning the whole brain. Example approaches are seed-based correlation and low-dimensional ICA decomposition [6]. In contrast to seed-based correlation and low-dimensional ICA, high-dimensional parcellation into many nodes (potentially hundreds) allows a richer analysis of the network connections; by shifting the emphasis from large-scale maps with fine spatial detail to a network description of nodes and edges, new information can be obtained. A detailed network model may reveal which nodes (sub-parts of the large-scale networks) are responsible for the correlations seen between the larger-scale networks [6]. To compare connectomes across subjects, it is essential

This work is supported by Pratiksha Trust Initiative in Brain, Computation and Data (Pratisksha Trust), India.

978-1-6654-9747-3/22/$31.00 ©2022 IEEE
DOI 10.1109/IPDPSW55747.2022.00043

to have the same parcellation in each subject, which preserves the node correspondence between the subjects. Once we apply a parcellation scheme in the form of a predefined atlas, we can assign each parcel (node) a representative time series, by averaging the time series from all the voxels within that parcel. We will thus obtain a $N_{\text{node}} \times N_{\text{timepoints}}$ data matrix, using which we can obtain a subject-specific $N_{\text{node}} \times N_{\text{node}}$ connectivity matrix. A major challenge for studying brain functional networks is to enable the application of biologically interpretable models using large numbers of nodes in a robust and practical way [6].

There are many studies dealing with the classification of brain functional networks generated using rs-fMRI images. In one such study by Jie et.al [7], sub-network kernel approach was used to compare similarity of brain graphs using the SVM tool. They have made use of ADNI data [8] for rs-fMRI images of subjects with progressive stages of Alzheimer's - normal (CN), mild cognitive impairment (MCI) and Alzheimer's disease (AD). The authors point out that the definition of brain nodes can dictate the accuracy of the classification task. Hence, this forms one of the topics under our study to enhance the methodology for improving accuracy. For any machine learning task like classification, it is always better to have large amount of training data. Since it is a tough ask to obtain a larger dataset under a single domain/disease, we have explored the capabilities of transfer learning [in section III] to pre-train the weights on some other similar, larger dataset and then train the models on the actual target dataset.

Graph Convolution Networks (GCNs) are neural network mechanisms that directly work on graphs and use techniques like convolutions and message passing to exploit their structural information [9]. GCNs have been increasingly used for classification of brain functional networks. It is important to analyze the capabilities and performance of the GCNs for the classification of brain networks. This will help in providing valuable insights on how and when the GCNs should be used for this domain. In this work, we have used GCNs for the classification of Alzheimer's data. We primarily use the Alzheimer's Disease Neuroimaging Initiative (ADNI) database, which contains brain images for different subjects for the purpose of classifying them based on the degree of progression towards Alzheimer's disease. As graph inputs to our GCN model, we use correlation matrices constructed using timeseries of regions of interest (ROIs) of the brain, obtained from the resting-state fMRI images [10].

We study the impact of the various parameters including the thresholds used for graph generation from the ADNI data, graph sizes, data sizes or the number of subjects and classification accuracy across different visits of the subjects. Secondly, to overcome the limitation of small data size in ADNI, we have used a transfer learning approach to initially train our model on a similar larger brain dataset and use these trained weights as initial parameters, and train on ADNI data. Finally, we have used GPU-based acceleration and CUDA streams to decrease the training time. The graph generation stage involves computing the Pearson Correlation Coefficient (PCC) across every pair of brain regions chosen as nodes and can be done in parallel. The amount of parallelism available increases as we increase the node count, and hence GPUs can be effectively used in graph generation. We have used CUDA streams for asynchronous computations of a batch for training while simultaneously transferring the data for the next batch. We find that the accuracy of the models improves when taking into account all visits of the subjects and also improves with increasing graph sizes. The use of transfer learning has also improved classification accuracy. Finally, the use of CUDA streams for asynchronous computations has resulted in decrease of execution times by up to 60%.

II. Data Setup and Preprocessing

A. Data Setup

The data for this work is downloaded from two publicly available brain data projects - Alzheimer's Disease Neuroimaging Initiative (ADNI) [8] and Autism Brain Imaging Data Exchange (ABIDE) [11]. Images for both databases are downloaded from the Laboratory of Neuroimaging's Image Data Archive (LONI IDA) [12], as resting-state fMRI images in the NIFTI format [13].

For our experiments, we primarily have made use of the data from the ADNI2 study. This study aims to understand the progression of Alzheimer's disease among individuals aged 55 to 90 years. Using images from a total of 189 subjects of mean age of 73.4 ± 7.4 years and multiple visits, a total of 609 images was obtained. The participants are divided into four categories depending on the mini-mental state examination (MMSE), Clinical Dementia Rating (CDR), and Memory Box score results, which indicate the progression of Alzheimer's disease. The scans were performed on Phillips Medical Systems scanners with a field strength of 3 T [14].

The raw data is arranged in the form of 4-D voxels, 3-D in space, and 1-D in time. Each voxel gives the BOLD (Blood

Oxygenation Level Dependent) reading, which is the brain activity at that particular voxel coordinate and time step. The preprocessing section outlines the steps used to create graphs from the raw data.

1) ADNI Data: The ADNI dataset classifies the images into four labels - CN (Control Normal), EMCI (Early mild cognitive impairment), LMCI (Late mild cognitive impairment), and AD (Alzheimer's disease). The images are collected from older population.

Since ADNI data contains information on the progression of Alzheimer's disease, multiple images are taken for each subject involved in the study over a period of time. Thus, our paper involves clinical data of 189 subjects with a total of 609 images. The demographic distribution of the 189 subjects is given in Table I.

Label	Count	Gender	Age
CN	52	30F/22M	75.59±6.76
EMCI	59	37F/22M	72.02±6.81
LMCI	45	18F/27M	72.91±8.39
AD	33	18F/15M	73.09±7.32

TABLE I: ADNI Subject demographic

B. Data Pre-processing

Preprocessing is a crucial step in preparing raw fMRI data for analysis. We aim to remove artifacts in the data, reduce noise, and use transformations to convert the images to a standard format. The preprocessing steps for this study involve the usage of FMRIB's Software Library (FSL) tools [15]. The images downloaded from the LONI IDA website [12] are raw fMRI images in the NIfTI format (Neuroimaging Informatics Technology Initiative) [13]. Following are the various preprocessing steps [16]. The steps are illustrated in Figure 1.

1) Step 1 - Brain extraction and smoothing fMRI images: FSL FEAT is used for fMRI analysis. We begin with the tool FSL BET for Brain Extraction, to remove the non-brain tissue including the skull and neck. Next, MCFLIRT is used for Motion Correction. Since the subject might not be perfectly still during the image acquisition, we align the images to the middle time-step(s) to account for random head movements. Subsequently, the FEAT tool also performs spatial smoothing using a 3D Gaussian kernel of full-width at half-maximum (FWHM) value of 5mm and intensity normalization of all time-series within a particular image. Also, the temporal high-pass filter of 0.01 Hz cut-off frequency is used to allow the stimulus readings to remain while removing noise and artifacts.

2) Step 2 - Linear Registration:: Subsequently, the images are aligned to a standard space using FSL-FLIRT (FSL's Linear Image Registration Tool) [17] where all the images are registered with the MNI152 template [18] (Montreal Neurological Institute's standard template based on the readings of 152 subjects) using 12 degrees of freedom. As a result, we obtain images of size 91x109x91 voxels for each time course. These T1-weighted images with 1mmx1mmx1mm voxels are consistent with the brain atlases used on the images.

3) Step 3 - Graph creation: Once the raw fMRI data is preprocessed, we obtain standardised images of size $91 \times 109 \times 91$ voxels for each time series. Thus each image is of size $91 \times 109 \times 91 \times 140$, since all ADNI images contain 140 time-steps. This is consistent with the ATLAS files we use to wrap the data and identify ROIs from the corresponding coordinates. We then generate graphs from the voxel data. The graph generation is explained in detail below.

Standardised image to adjacency matrix: We primarily make use of the library NiBabel [19], which is created for the purpose of neuroimaging in Python. We make use of the AAL (Automatic Annotated Labelling) atlases, since the AAL atlases are defined in the same standard space as our data (MNI152). We make use of the light AAL90 atlas which specifies 4132 voxel coordinates and their corresponding ROI (Region of Interest) label.

We begin by loading the data using NiBabel into a NumPy array, and the AAL90 atlas in the same format, and map each voxel of the atlas to the corresponding voxel in the image data array along with its corresponding label. We take the average of the BOLD value of all voxels within an ROI and generate a time-series matrix for each image of size 90 $\times$ timesteps. Then, we normalize each time-series matrix, so that the values across images can be compared. From the normalized timeseries matrix, we generate a correlation matrix by calculating the Pearson correlation coefficient (PCC) of BOLD time-series values of each pair of ROIs.

The GCN model we are currently implementing makes use of adjacency matrices as a graph input. Thus, using appropriate cut-off value to threshold the PCC values in the correlation matrix, we binarize it and obtain an adjacency matrix as a graph input. For our work, we used the cut-off value of 0.15.

Graph coarsening: Graphs that are generated keeping 4132 voxels $\times$ 140 timesteps will lead to 4132-node graph. We have

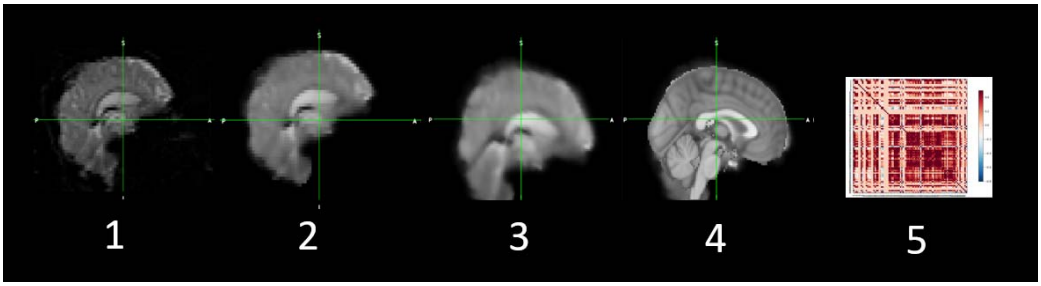


Fig. 1: Preprocessing the raw images: (1) Raw fMRI image (2) Brain extraction and motion correction (3) Linearly registered with MNI152 space (4) Comparison of the preprocessed image with standard image (5) Correlation matrix generated by PCC of time-series matrix

explored the options for coarsening the graph to generate a smaller graph of about 200 vertices, without compromise on the meaningful representation of graphs. The procedure for graph coarsening is as follows.

We partition the graph into subgraphs, each containing nodes from one region of interest (ROI) in the atlas. Each ROI is represented by one subgraph. Using Louvain's algorithm for community detection [20], we form communities/clusters from each subgraph. Each community so formed is considered as one node of the coarsened graph. We generate the edges in the coarsened graph as follows. One node in the coarsened graph consists of many vertices of the original graph. we define an edge between two nodes of the coarsened graph if there were more than a certain fraction of all the possible edges between each pair of vertices from two different communities. For our work, we used a fraction of 0.5. We ignore the self edges, i.e., the edges between vertices belonging to the same community.

III. Hierarchical GCNs and Transfer Learning

A. Baseline GCN Model

For the baseline model, we used Standard Graph Convolution Networks (GCNs). Feedforward networks are best suited for regular and euclidean data, but graph structures that are non-Euclidean requires networks that can capture the structural information.

GCNs use the message passing technique where the model learns the node features by inspecting and aggregating features from neighbouring nodes [9]. Formally, the GCN take two matrices as input - 1) Feature Matrix X of size $N \times D$, where N is the number of nodes and D is the number of Input features and 2) Adjacency Matrix A of size $N \times N$ and generate Graph Representations Z as output. The intermediate hidden layer can be written as a non-linear function

$$H^{l+1} = f(H^l, A) \tag{1}$$

$$H^{(0)} = X \tag{2}$$

$$H^{(L)} = Z \tag{3}$$

where L is the number of layers. The function $f(.;.)$ includes enforcing self-loop in A, normalizing it by the degree matrix, D, and passing the arguments to feed forward network with ReLU activation function, we get

$$H^{l+1} = f(H^l, A) = ReLU(\hat{D}^{\frac{-1}{2}}\hat{A}\hat{D}^{\frac{1}{2}}H^l W^l) \tag{4}$$

$$\hat{A} = A + I \tag{5}$$

where I is the identity matrix and $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$.

At the last layer, the learned node embeddings are averaged to get the graph embedding. Although workable, the graph level representation generated using this method is inherently "flat", since the entire graph structure information is neglected during this process. Also, averaging large number of node embeddings leads to substandard graph embeddings.

B. Hierarchical GCN

Hierarchical Graph pooling with structured learning [21] is used for all our runs. This model learns graph representations in a hierarchical way with two operations at each layer: (i) **Node representation learning** by message passing technique,

similar to Baseline GCN model and (ii) **Structure learning** downsampling by node clustering.

Hence by the end, the essential topological information is preserved. At each layer, two neural network models are used: (i) GCN model for node representation and (ii) A Neural Network to cluster similar nodes, to coarsen and learn the structure of the graph. The graph pooling operation proceeds by selecting a subset of the nodes at each layer to form a subgraph for the subsequent layer. This method is better suited for our study since the brain data can be expressed as graphs, and hence GCNs are suitable.

Figure 2 illustrates the working of the hierarchical GCN model. The effectiveness of this method had been tested by Zhang et.al [21] on publicly available datasets, where they have demonstrated better classification accuracies compared with the existing kernel, GNN, and the pooling-only strategies for these datasets.

C. Transfer Learning

Transfer learning is a ML method where a model trained for a task is reused as the starting point for a model on a different task, assuming both the tasks are sufficiently similar. This helps in the utilization of knowledge (features or weights) acquired from source learned to solve a related target task. The benefits of transfer learning are : (i) **Higher Start:** The initial learning of the model, prior to training is higher than it otherwise would be, (ii) **Higher Slope:** The learning rate is steeper, this convergence is faster, and (iii) **Higher asymptote:** The convergence of the model is at higher performance level.

These benefits facilitate the model to perform well even in low data regime. In our experiments, the target dataset - ADNI has a limited size (189 subjects). Thus to improve the model's performance, we pre-trained our model on a larger dataset, called ABIDE (Autism Brain Imaging Data Exchange). ABIDE has a dataset size of about 600 subjects. We trained the source model on the ABIDE dataset, the weights are saved and the same weights are used to initialize the target model while training the ADNI dataset.

IV. High Performance Machine Learning

We also used high performance computing approaches for the acceleration of some of the steps described earlier.

A. Graph creation with GPUs

As explained earlier, we have to compute Pearson correlation coefficient(PCC) for 4132 $\times$ 4132 voxel pairs. In general, we will have to compute PCC for $N \times N$ voxel pairs where N can be quite large, which is time consuming, and hence suitable for HPC acceleration. We have used GPU kernel calls to generate adjacency lists for each node, in parallel. We pass the entire voxel $\times$ timesteps matrix into the GPU device memory. In each kernel call, we select one row (read voxel) at a time, and we compute PCC of this row with all other rows of the matrix and thus obtain a list of correlation values corresponding to that row. After applying a suitable threshold, this list is converted to adjacency list which can be transferred back to host memory and stored. After completing the kernel calls for all the rows, we will have the adjacency matrix of the graph ready. We have enabled the code to run for upto 95K voxel entries, which can be scaled further, depending on the application scenario. For 4132 node graph, the performance of pandas based python code and our GPU code had similar runtimes (about 8 seconds). This is due to lesser availability of data parallelism and lower node count. When we tested with greater number of vertices (95k), our GPU acceleration for the graph generation step resulted in the reduction of execution time from tens of hours to a few minutes (about 25min).

B. GPU Executions for Aysynchronous Computations in GCNs

The HGP-SL GCN model implemented in this paper makes use of the GPU PyTorch library in Python for Deep Learning. The model is trained and tested successively over several epochs and validated at the end. For each iteration of training and testing, the data stored in the host (CPU) is communicated to the device (GPU).

We begin by performing an analysis of how the total time taken is being distributed among different operations. We notice that the majority of the time is taken during the training (around 84 %) as shown in Figure 3a. Upon further splitting the training step, we notice that the majority of the time is consumed for transferring the data from the CPU to the GPU at every iteration, as shown in Figure 3b.

Thus, we conclude that the data transfer from host to device poses a significant bottleneck, as the subsequent operations of training the model can only be resumed once the data has been received by the GPU. As our dataset grows in size, either due to increasing graph sizes or number of images, this causes a bottleneck in both the times taken and memory consumption.

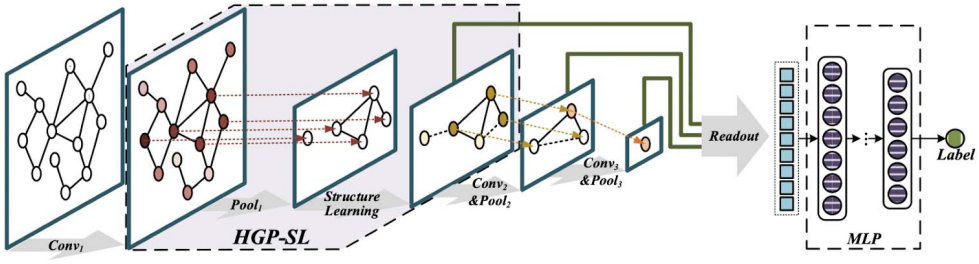
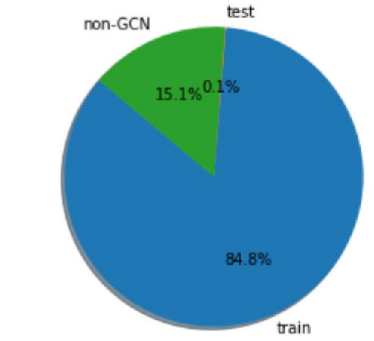
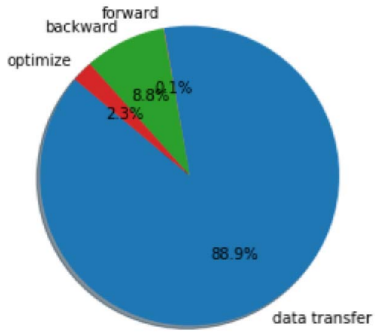


Fig. 2: HGP-SL model [22]



(a) Split-up of total time taken



(b) Split-up of training time alone

Fig. 3: Time split-ups

We make use of CUDA streams to hide the data movement overheads. Instead of sequentially proceeding from training to testing during each iteration on the default stream, we introduce an additional stream only for data transfer of the next iteration's batch of data. Thus, instead of all the processes being run on the default stream on the GPU, we implement two parallel CUDA streams; one for the training of the present iteration, and the other for the data transfer of the succeeding iteration. The preceding iteration's data is erased from the GPU once it is used for training the model. For this, we have implemented bookkeeping of the regions of the memory used by the CUDA kernels.

Thus, using asynchronous computation by utilizing two parallel CUDA streams and conducting data transfer and training at the same time help to tackle the bottleneck that the data transfer poses. The results of this implementation are demonstrated in the next section.

V. Experiments and results

Unless explicitly specified, for each run we use 200 epochs, batch size 4, learning rate 0.01, dropout ratio 0.5. The training set is 70% of the total dataset, chosen randomly. All other parameters are the default as set by the code. For ADNI data with 609 images from each subject, the labels are AD - 92, MCI - 363, CN - 154.

For each run, we used 70% of the subjects for training, 20% for testing and 10% for validation. The split was performed before running the GCN so care was taken that the same subject was not used for training, testing and validation. The same splits were also used for transfer learning. Hi-GCN was trained on our own (ADNI) data and the results mentioned correspond to these experiments. We used the same dataset

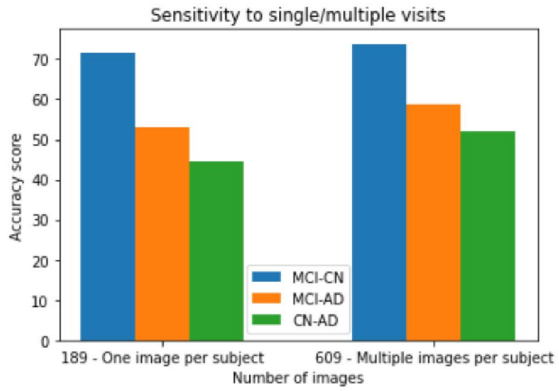


Fig. 4: Comparison of accuracy scores between single visit and multiple visits of the subjects.

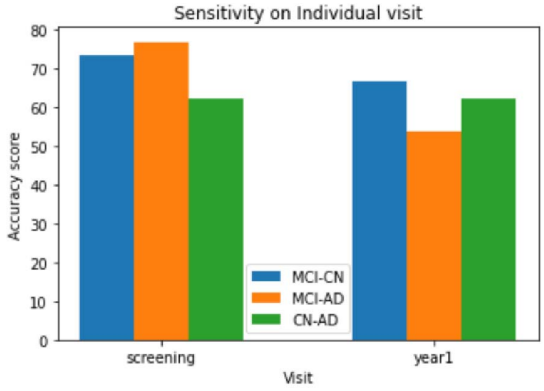


Fig. 5: Comparison of accuracy scores at the time of screening and after 1 year of screening for different subjects.

(ADNI) and similar split (70/30/10) as we used with the HGCN model.

First, we show results on the effects of various parameters on the classification accuracy of the GCN models. Next, we show the effectiveness of the transfer learning approach. Finally, we show the reduction in execution times due to HPC acceleration.

A. *Effect of Multiple Visits*

Working with the ADNI dataset, we analyse the effect of multiple visits for each subject on the accuracy of the model. For this purpose, we consider the last visit of each subject and make our analysis. From Figure 4, we observe that accuracy scores improve with multiple visits. This is mostly due to the availability of a larger number of training images for each label.

B. *Classification Accuracy for Visits of Same subjects*

The available subject data is pre-grouped into separate sets for each visit of a subject. Here, we consider the images of the common subjects across the screening visit and the visit one year later, which gives us 81 common subjects in both subsets. The label information for these subjects are CN - 20, MCI - 47, and AD - 14. We try to understand how the classification accuracy is impacted over time.

From Figure 5, we can observe that accuracy scores involving MCI, esp MCI vs AD, have dropped from screening to year 1. This might be attributed to the fact that with the progression of the disease, MCI subjects might become more similar to AD, and thus resulting in reduced classification accuracy.

C. *Effect of Graph Sizes*

Here, we compare graphs created using two atlases - AAL90 and AAL116. As labeled in Figure 6, *AAL90-90* uses the AAL90 atlas as-is, *AAL90-196* uses graph coarsening method described in Section II-B3 to split the entire atlas into 196 regions. In other words, each of the 90 ROIs is partitioned into about 2-3 sub-regions and edges are defined across each pair of sub-regions. *AAL116-116* uses the AAL116 atlas as-is and *AAL116-580* uses k-means clustering to divide each ROI in the AAL116 atlas into 5 regions.

We notice that the accuracy scores across all graph sizes are similar for MCI vs CN. But a significant jump in accuracy is observed for the other two classification tasks when graph coarsening methods were applied for AAL90 atlas defined ROIs to obtain a larger 196v graph. However, the application of K-means clustering on AAL116 does not improve accuracy since it does not take into account the naturally defined sub-regions in the brain, as it predefines the number of sub-regions expected from each region. The graph coarsening method, which is based on the activity happening across and within each region, to sub-divide the regions, shows good improvement in the accuracy score.

D. *Varying Threshold Cut-offs*

Each subject's fMRI image is processed to generate correlation matrix, which is converted into an adjacency matrix

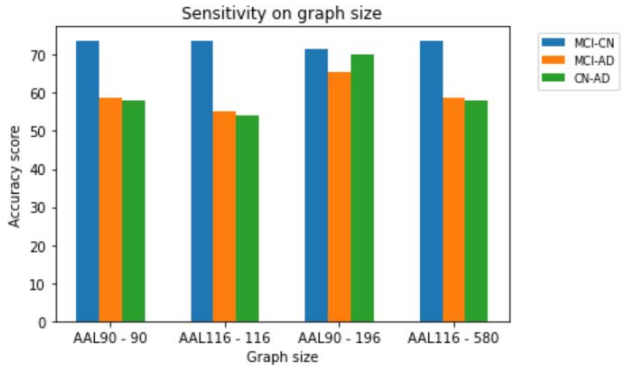


Fig. 6: Comparison of accuracy scores across graphs of different sizes generated by using different brain atlases.

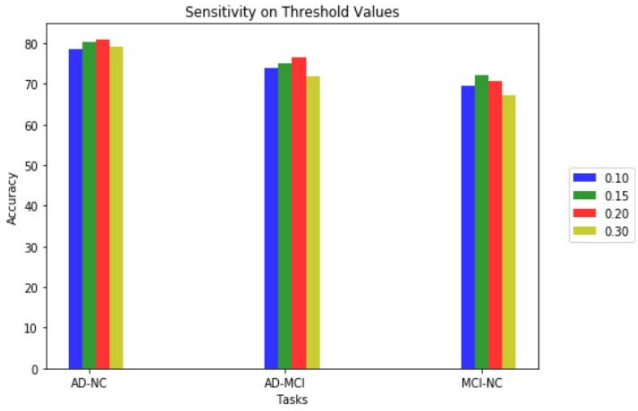


Fig. 7: Comparison of accuracy scores across different threshold values. Adjacency matrices are generated by applying threshold values on the correlation matrices.

by applying a threshold on the PCC values. Negative PCC values are converted to positive by applying mod function. To choose the ideal threshold value, we analyze the effect of different thresholds on accuracy. Results are shown in Figure 7.

We observe that increasing the threshold value leads to sparser graphs which resulted in lesser classification accuracy; on the other hand, low threshold values lead to highly dense graphs that will deplete the meaning of functional connectivity of the regions (nodes). Thus, a trade-off is to be maintained. Threshold of 0.15 relatively performs better in MCI vs CN, while 0.2 performed marginally better in AD vs CN and AD vs MCI. So a threshold value between 0.15-0.2 can be chosen to give the best results.

E. Transfer learning with ABIDE data

The ADNI dataset has a limited number of unique subjects (189), and hence is not quite appropriate for employing deep learning algorithms. Thus, we make use of another resting-state fMRI dataset, the ABIDE dataset [11], which is centered on autism subjects and has 589 subjects. The ABIDE dataset classifies the images into 2 labels - CN (Control Normal) and AU (Autistic). The images are collected from a younger population. The demographic distribution is given in Table II.

Label	Count	Gender	Age
CN	300	56F/244M	18.24±8.19
AU	289	29F/260M	17.67±8.93

TABLE II: ABIDE Subject demographic

We use the weights of the model pre-trained with the larger ABIDE dataset to initialize the smaller ADNI dataset. From Figure 8, we see that employing transfer learning has marginally improved the accuracy scores of the 3 ADNI classification tasks. A possible reason for not obtaining much better improvement is that i) ADNI dataset has 3 labels and ABIDE has 2. Thus, it would only partly help to pre-train the GCN weights with ABIDE, to be later used by ADNI. A future task could be to use a dataset with 3 labels similar to ADNI. ii) There is a dissimilarity in the datasets with respect to the age group; ABIDE dataset has a mean age of around 18yrs, while ADNI dataset has a mean age of around 73yrs. This affects the similarity in the rs-fMRI images, even among normal controls, and hence would contribute in reduced improvement in accuracy scores due to transfer learning.

F. Comparisons with State-of-Art

We compared our models with another GCN-based brain disorder prediction model Hi-GCN [21] proposed by Jiang et al and with our baseline standard GCN model, SGCN. SGCN is the standard GCN approach, where the model learns embedding for each node using the message passing technique and the last layer is a pooling layer to get the embedding for the complete graph. HGCN also learns the embedding of the graph in a hierarchical way, but has two operations at each layer, namely, graph pooling and structure learning. The graph pooling operation utilizes node features and graph structure

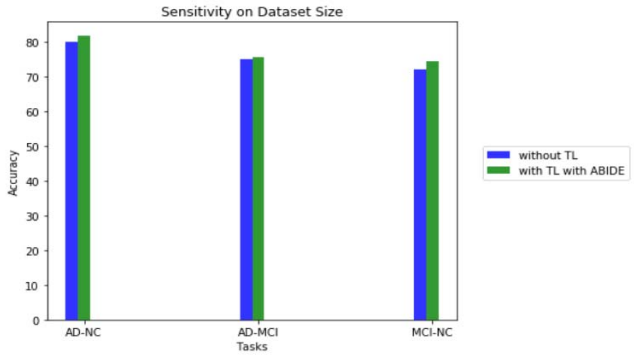


Fig. 8: Comparison of accuracy scores between models with and without using Transfer learning with ABIDE dataset.

information to perform down-sampling on graphs. Then, a structure learning layer is stacked on the pooling operation, which aims to learn a refined graph structure that can best preserve the essential topological information.

Our approach, Hi-GCN is a hierarchical version of GCN, but the hierarchy is defined in a different way. It jointly learns the graph embedding taking care of two aspects, namely, region to region brain activity correlations in the same subject and subject to subject relation in the population network (all graphs). So, a high-level embedding of brain network representation is learned by deriving the embedding for a single subject and aggregating embedding of its neighboring subjects in an end-to-end fashion with global supervision.

Time complexity is less for SGCN and similar for Hi-GCN and HGCN. Table III summarises the classification accuracy results for AD Vs MCI. The hierarchical GCN model with Transfer Learning (HGCN w/ TL) performed better than the other models. This can be attributed to a better hierarchical encoding of the brain graphs and use of prior knowledge via transfer learning.

G. Asynchronous Computations with CUDA Streams

In this subsection, we analyze the effect of CUDAstreams on the total runtime of the model. We perform the comparison with the same datasets with graphs of 196 nodes.

Figure 9 shows the execution times with and without using CUDA streams. We notice that as the batch size increases, the effect of CUDA streams becomes more evident, becoming half the runtime of executions without CUDA streams towards larger batch sizes of 128 images. In the case of graphs with 196 nodes, our approach conserves memory by consistently deleting unnecessary data during each iteration of training and testing by bookkeeping of memory regions. Thus, we find that CUDA streams enable us to run our model with batch sizes of 128 images even when it runs out of memory in the original flow.

Model	**AD vs MCI (Average Accuracy)**
Hi-GCN [21]	78.5
SGCN	80.0
HGCN	80.3
HGCN w/ TL	**81.8**

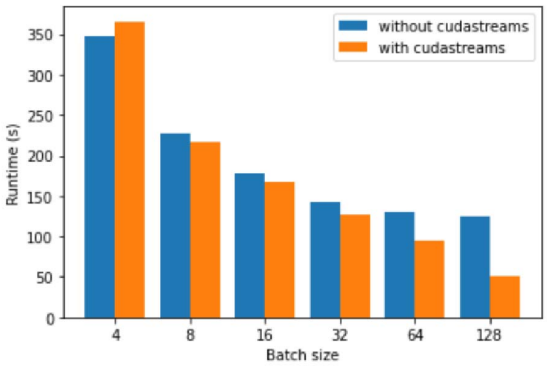
TABLE III: Classification accuracy across different models. Hi-GCN is the model proposed by Jiang et al. SGCN is our baseline standard GCN model, HGCN is the hierarchical model w/ and w/o Transfer Learning (TL).

VI. Conclusions and Future Work

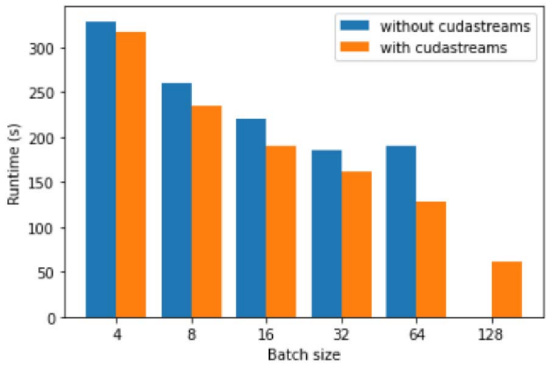
In this paper, we have explored the robustness of GCN models for classification in brain functional networks obtained via rs-fMRI and we have analyzed the effect of various parameters including number of visits, number of subjects, size of brain graph constructed and dataset sizes. We have performed this study with Alzheimer's data. We have also proposed a transfer learning approach to use large-scale data to train models and use the models for classification in networks where the data size is comparatively small. This is important to study different brain networks where the data may be sparse. Finally, we have proposed HPC acceleration methods including use of CUDA streams to reduce the training time.

We used two GCN based methods - standard and hierarchical, for the MCI-AD classification. The use of transfer learning has also improved classification accuracy. Finally, the use of CUDA streams for asynchronous computations has resulted in reduction in execution times by up to 60%.

We would like to extend this study for different kinds of brain networks. We also plan to develop domain adaptation techniques to improve the accuracy of the models even further. Finally, we also plan to develop GPU parallel methods for computations of similarities between the different brain networks. In particular, methods will be developed to accelerate GED (Graph Edit Distance) computations [23], for comparing brain networks.



(a) 90 nodes



(b) 196 nodes

Fig. 9: Effect of cudastreams on runtime comparison

REFERENCES

[1] R. K. Lama and G.-R. Kwon, "Diagnosis of Alzheimer's Disease Using Brain Network," *Frontiers in Neuroscience*, vol. 15, p. 15, 2021. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2021.605115

[2] J. Zhou, S. Liu, K. K. Ng, and J. Wang, "Applications of Resting-State Functional Connectivity to Neurodegenerative Disease," *Neuroimaging clinics of North America*, vol. 27, p. 663–683, 2017.

[3] H.-J. Park and K. Friston, "Structural and Functional Brain Networks: From Connections to Cognition," *Science*, vol. 342, no. 6158, 2013. [Online]. Available: https://science.sciencemag.org/content/342/6158/1238411

[4] F. V. Farahani, W. Karwowski, and N. R. Lighthall, "Application of Graph Theory for Identifying Connectivity Patterns in Human Brain Networks: A Systematic Review," p. 585, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2019.00585

[5] D. C. Van Essen, S. M. Smith, D. M. Barch, T. E. Behrens, E. Yacoub, and K. Ugurbil, "The WU-Minn Human Connectome Project: An overview." pp. 62–79, 2013.

[6] S. S. et al., "Functional Connectomics from Resting-state fMRI." p. 666–682, 2013.

[7] B. Jie, M. Liu, D. Zhang, and D. Shen, "Sub-Network Kernels for Measuring Similarity of Brain Connectivity Networks in Disease Diagnosis," *IEEE Transactions on Image Processing*, vol. 27, no. 5, p. 2340–2353, 2018.

[8] "ADNI (Alzheimer's Disease Neuroimaging Indiative)," http://adni.loni.usc.edu/.

[9] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," 2017.

[10] T.-A. Song, S. R. Chowdhury, F. Yang, H. Jacobs, G. El Fakhri, Q. Li, K. Johnson, and J. Dutta, "Graph convolution neural networks for alzheimer's disease classification," 2019.

[11] "ABIDE (Autism Brain Imaging Data Exchange)," http://fcon_1000.projects.nitrc.org/indi/abide/databases.html.

[12] "LONI Image Data Archive," https://ida.loni.usc.edu/login.jsp?project=ADNI&page=HOME#.

[13] "NIfTI (Neuroimaging Informatics Technology Initiative)," https://nifti.nimh.nih.gov/.

[14] Y. Kazemi and S. Houghten, "A deep learning pipeline to classify different stages of alzheimers disease from fmri data," *2018 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, 2018.

[15] "FSL," https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FSL.

[16] S. Sarraf, D. D. Desouza, J. Anderson, and G. Tofighi, "Deepad: Alzheimer's disease classification via deep convolutional neural networks using mri and fmri," 2016.

[17] "FSL-FLIRT," https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FLIRT.

[18] "MNI152," http://nist.mni.mcgill.ca/icbm-152lin/.

[19] "NiBabel," https://nipy.org/nibabel/.

[20] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast Unfolding of Communities in Large Networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, Oct 2008. [Online]. Available: http://dx.doi.org/10.1088/1742-5468/2008/10/P10008

[21] H. Jiang, P. Cao, M. Xu, J. Yang, and O. Zaiane, "Hi-GCN: A Hierarchical Graph Convolution Network for Graph Embedding Learning of Brain Network and Brain Disorders Prediction," *Computers in Biology and Medicine*, vol. 127, p. 104096, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482520304273

[22] Z. Zhang, J. Bu, M. Ester, J. Zhang, C. Yao, Z. Yu, and C. Wang, "Hierarchical Graph Pooling with Structure Learning," 2019.

[23] X. Chen, H. Huo, J. Huan, and J. S. Vitter, "An Efficient Algorithm for Graph Edit Distance Computation," *Knowledge-Based Systems*, vol. 163, pp. 762–775, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095070511830488X