Applications of Connectivity in the Human Brain Using Functional MRI data: Classification with Graph Neural Networks

Andrew Cheng
apcheng@ucsd.edu

Daphne Fabella
dfabella@ucsd.edu

Terho Koivisto
tkoivisto@ucsd.edu

Daniel Zhang
yiz029@ucsd.edu

Gabriel Riegner [*]
gariegner@ucsd.edu

Armin Schwartzman [*]
armins@health.ucsd.edu

Abstract

In our project, we developed classification models to discern gender difference from functional magnetic resonance imaging (fMRI) to examine the applicability of Graph Neural Networks (GNN) on resting state fMRI. By exploring brain network differences and use of data simulation with controllable parameters, we constructed a neural network classifier utilizing graph structure alongside a simple baseline model. The results show the two classes are separable when only comparing fMRI resting state data. Our research will walk through steps of statistical methods, simulation, and model description to accomplish our objective.

Code: https://github.com/AndrewCheng02/DSC180B-Capstone-ProjectA09

[*] Capstone mentor

1 Introduction

Functional magnetic resonance imaging still proves to be a key tool to deciphering the mysterious architecture of the brain after its first uses in the early nineties. Due to the high spatial resolution (in millimeters) and relative ease of getting a scan, data for fMRI scans has become highly available and useful. These scans still lack temporal resolution due to how fMRI blood-oxygen-level-dependent (BOLD) signals work, where it is the blood flow that is being measured rather than the signals in the brain. This will make our temporal resolution in seconds rather than some other scanning methods (e.g. MEG, EEG, NIRS) that are in milliseconds. Understanding and simulating fMRI data is crucial for understanding the connectivity of the brain.

Our objective is to create a classifier which captures patterns in resting state connectivity to predict the gender label of the subjects. The purpose of the task is meant to serve as a proof of concept to apply Graph Neural Networks (GNN) on resting state fMRI readings. While our task is to predict gender, as more labeled data comes available the prediction task can be changed to a more traditional medical field use of fMRI and predict neurological disorders. For now using gender as the label should showcase the ability and sensitivity of our classifier to be suited for the task.

Our project will focus on data from the Human Connectome Project (HCP) Young Adult data release, which has a collection of 1,200 fMRI scanned young adult brains. The data is in the form of voxels, which are pixels that exist in a three-dimensional space of a brain mapping to an intensity of a BOLD signal at a specific point in time. With the amount of different brains, it would be impossible to map one to one comparisons between specific voxel locations in the brain. So we use pre-processed data that shrinks the number of subjects to 1,003 subjects. We will later discuss the data in more detail.

We are not the only ones to be interested in the differences of fMRI data in resting state between genders. A study by Sie et al. (2019) investigated age and gender differences in resting state functional connectivity. They found during rest females will have stronger correlations in the default mode network (DMN) than their male counterparts. While these differences were shown to wear off with age, this still proves there exist possible indicators to be found in fMRI data.

Our main model, the GNN classifier, uses graphs as input which is suitable for predicting a label from fMRI data after it has been pre-processed to a correlation matrix. The GNN is built on PyTorch Geometric, an extension to PyTorch, which has large libraries of functions designated for GNNs. Using a GNN for this type of task is also a relatively new feat, as the type of GNN we are using, the Graph Convolutional Network (GCN), was published in 2016 by Kipf and Welling (2016). GCNs work similar to CNNs which have a sliding filter, but instead of analyzing by proximity, GCNs update via connected neighbors. Given a correlation matrix we can use the data in a graph structure and feed it to our model and output a prediction.

Since we want to have a baseline to the outputs of the GNN, we also used a k-nearest neighbor (k-NN) from the python sklearn library. The k-NN models don't analyze graph structure, but treat a correlation matrix as a long array of features. Then uses distance to place the most similar arrays together, and very different arrays far apart. The result attempts to make the data linearly separable where it's easy to predict the label of the subject.

For our GNN and k-NN, we must explain some important considerations. For data preprocessing we explain changes we need to make to data. Simulate an environment to prove the classification is practicable from correlation data. Then explain our exploratory data analysis to display significant differences in the data between genders. Finally analyze the results of our models to see the performance for resting state fMRI classification.

2 Background

2.1 HCP Data

HCP is a research initiative that aims to map the structure and connections in the brain through fMRI. It started in 2009, and the project has become a joint effort involving various institutions in the United States. Our data comes from the HCP young adult study which released 1200 young adult subjects in 2018. The subject's data has been preprocessed to contain the denoised transformed network matrices and time series for each subject. The data also includes many features such as age and gender that will be useful in building our classification models. The data also includes an Atlas which is a choice of mapping to assign the brain regions. The HCP uses the an ICA decomposition atlas; later described in depth in the next section. Having the atlas allows us to train other fMRI scans outside of HCP on our model by transforming any Nifti file (fMRI 4D raw data) into a network matrix with the same preprocessing methods as the HCP data.

2.2 PCA and ICA

Raw fMRI data is a 4D matrix consisting of over 92,000 voxels in 91 by 109 by 91 space over a time frame. We need a reduced version of the dataset to efficiently compute statistical methods and analyses and one of the ways we can group fMRI data is by decomposing the data into different components. This is called multivariate decomposition, where the data we observe comes from a number of separate sources that have been mixed together. We can then use factorization methods to identify these components and create regions of the brain that reduce the dimension of our data. The next two methods are commonly used to pinpoint areas of interest in the brain.

The first factorization method is Principal Component Analysis (PCA). PCA finds the orthogonal eigenvectors of the covariance matrix, otherwise known as the principal components. For example, given a set of points in 2D space, the first principal component is the direction along which the data has the most variance. The second principal component is the direction that explains the next greatest amount of variance and is uncorrelated with the first principal component. PCA has the benefit of being easy to understand and to implement. It can also be used for data reduction, where we only perform analysis on the principal components instead of thousands of individual data points (voxels) of the fMRI scan. However, as PCA is only sensitive to Gaussian data, it is not useful for many of the signals in fMRI data that do not follow a Gaussian distribution.

On the other hand, Independent Components Analysis (ICA) is able to find the components only if the data is not orthogonal (independent variables of the dataset are correlated) nor Gaussian. This factorization method was created to solve blind source separation problems, where signals from multiple sources have been mixed together, and the goal is to identify the different sources. The model for ICA is defined as $X = As$, where X is the data we observe, s is the set of components we are trying to identify, and A is the mixing matrix that combines the components. By assuming that the individual components are statistically independent, the resulting combination is likely to be non-Gaussian and able to be decomposed with ICA.

To begin preprocessing the HCP data, we first apply PCA to reduce the overall dimensionality of the data. PCA performs better than ICA when reducing large datasets since it's computationally faster and simpler with out having a mixing matrix. In addition, a large dataset guarantees the data to become more Gaussian according to the Central Limit Theorem. After reducing the data through batches of PCA, the final step is reduced through ICA. The HCP database contains ICA decomposition for 15, 30, 100, 200, and 300 dimensions. For this report we will focus on using the 100 dimension data.

2.3 Correlation

The input for our model will be the pairwise correlation matrix generated by the fMRI time series also known as the network matrix. A common approach for estimating functional connectivity is finding the correlation values between two brain regions' activity over time. The fMRI scans used to generate the HCP data are preprocessed and analyzed to provide 4,950 pairwise correlations found via PCA and ICA for each subject. The 4,950 pairwise correlations are generated from the 100 regions choosing each other:

$$\binom{100}{2} = \frac{100 \cdot (100-1)}{2} = 4,950$$

When analyzing a dataset encompassing more than two brain regions, a correlation matrix can be employed to explore the linear relationships among all possible pairs. We perform a series of analyses on these correlation matrices to explore the possible distinctions between the functional connectivity of the male brain and the female brain.

*Note that the correlation matrix is symmetric and has an empty diagonal

2.4 Partial Correlation

One major issue with pairwise correlations is that regions can be indirectly correlated with each other. For example, if X and Y are correlated to Z, then X and Y would have correlation with each other in the pairwise correlation matrix despite having no direct connections. To solve this problem we use Partial Correlations. A Partial Correlation matrix is generated by regressing the regions against every other region when before computing the correlation. In this case the instrument or confounding regions are accounted for which reduces the amount of false correlations. The formula for generating a partial correlation can be found at this link or in the notebooks.

2.5 Fisher Transform

The correlation matrix calculated from the fMRI scans undergo a Fisher Transformation. This Fisher transformation, which is achieved by taking the inverse hyperbolic tangent of the original Pearson correlation coefficients, amplifies pairwise correlations as they approach 1 or -1 and yields a new variable that has an approximately normal distribution.

The variable's normal distribution and amplification of highly correlated regions facilitates parametric testing in later sections reveal interesting patterns and statistically significant differences in BOLD signals between the male and female brain.

Fisher Transformation formula:

$$f(x) = arctanh(x)$$

3 Methods

3.1 Data Generation

We decided to first test our classification methods on simulated data. To do this, we need to generate data with simple relations between variables that would clearly show on a correlation matrix.

We start by generating an $n \times m$ matrix A, where n is the number of time points and m is the number of regions in our simplified model of the brain. The greater the number of time points, the less noise will appear in the correlation matrix. Each cell in this matrix is filled with Gaussian noise. This simulates the different regions of the brain and their activity at each time point.

Box 1 *Simulate Brain Activity with Gaussian Noise*

$$A_{n,m} \overset{\text{iid}}{\sim} N\left(\mu, \sigma^2\right)$$

where

A = *generated dataset of brain activity*
μ = *mean of noise*
σ^2 = *variance of noise*

We want to simulate the connections between brain regions. When one region of the brain becomes active, there will also be activity in the other areas of the brain that are connected to it. This connection between the regions can be seen in the correlation matrix.

To mimic this, we first separate the matrix A in to two sets of rows, the active set B and inactive set C. This split is determined by a proportion p, ie. if $p = 0.2$, then B is the first 20 percent of rows and C is the remaining 80 percent.

We then define a $1 \times m$ vector of weights $\overrightarrow{v}$, which is preset beforehand, and define the matrix w as the diagonal matrix of the weights $\overrightarrow{v}$. To get the final result, we multiply the active set B with the matrix w then concatenate the product with the inactive set C.

Box 2 *Correlate Brain Regions with Weights*

$$\overrightarrow{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}$$

$$w = Diag(\overrightarrow{v})$$

$$result = \begin{bmatrix} Bw \\ C \end{bmatrix}$$

where

$\overrightarrow{v}$ = *vector of weights*
w = *diagonal matrix of weights*
$result$ = *final processed generated data*

This process forces regions of the brain to be correlated with each other according to the vector of weights. For example, if we want regions a and b to be connected, we can set their weights to 2 while the other regions have a weight of 1. This means that for some time points, an increase in region a will correspond to an increase in region b, which will cause the two regions to be positively correlated.

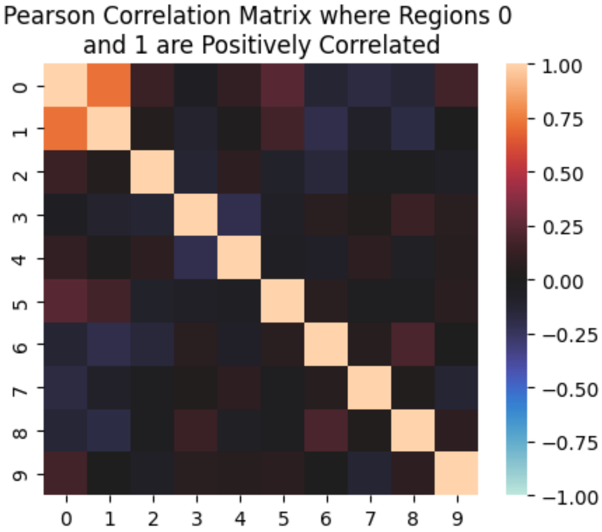


Figure 1: Pearson correlation matrix with weights: $[2, 2, 1, 1, 1, 1, 1, 1, 1, 1]$

We now need to define the proportion of the initial split p, which determines the amount of active and inactive time points. The goal of this method is to force regions of the brain to correlate with each other, so a value of p that results in a large correlation coefficient is preferable. To find a suitable p, we calculate the average correlation of the resulting matrix for values in its range.

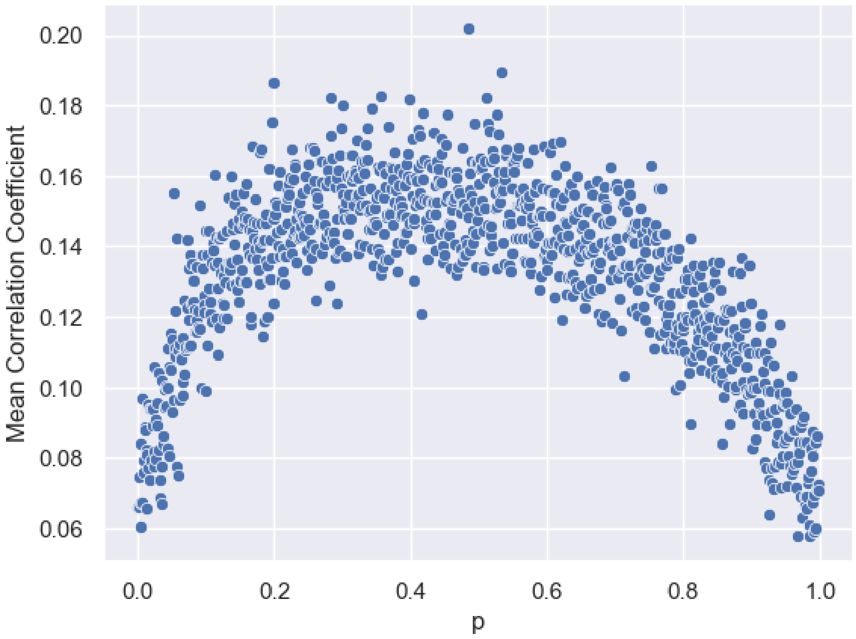


Figure 2: Proportion of the active set vs mean correlation coefficient. Each point is a simulated subject generated with the specified proportion.

From the plot above, we can see that as p increases, the correlation first increases and then decreases, being at its lowest at p values of 0 or 1. This makes intuitive sense, as this is when either all the time points are active or none are. We need both active and inactive sets to calculate the correlation between brain regions. If none of the time points are active, then the data is only Gaussian noise. If all of the time points are active, then the weights simply become inherent features of the brain regions. The value of p for when the correlation is most significant is around 0.4, which is what we ended up choosing for the proportion.

This process works well for generating Pearson Correlation matrices. However, Partial Correlations exclude other features when calculating the correlation between two regions. This means the more regions that are forced to be correlated with the weights in a single vector, the less visible the correlation will be on a Partial Correlation matrix.

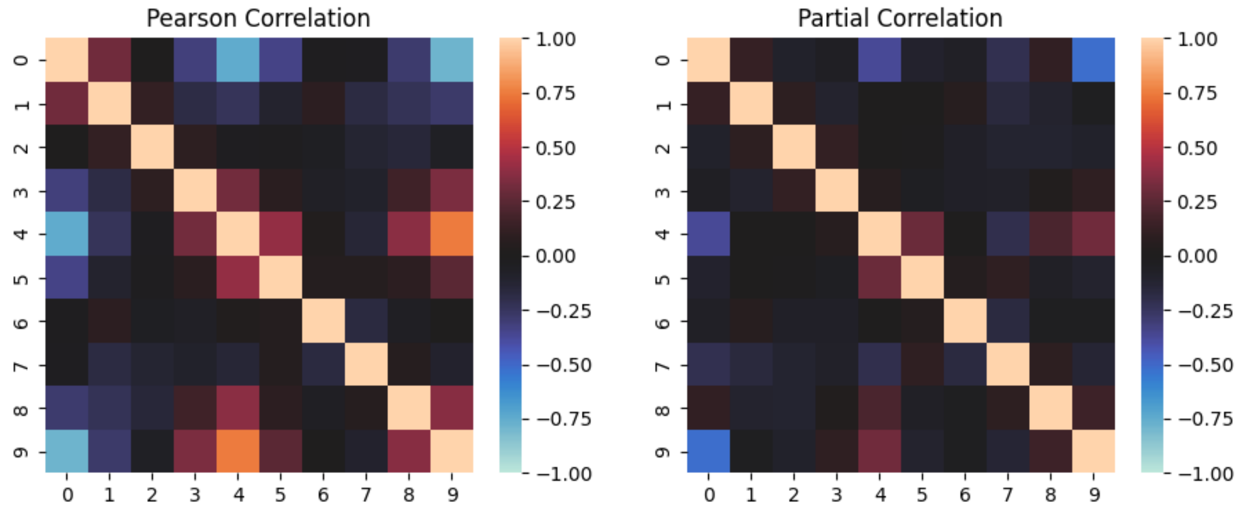


Figure 3: Pearson vs partial correlation with weights: $[0.4, 0.8, 1, 1.2, 2, 1.2, 1, 1, 1.2, 2]$

To fix this, we repeat this process with multiple different weights, shuffling and recreating the active and inactive sets each time. This way, we can model more complex relationships than we could with a single vector of weights.

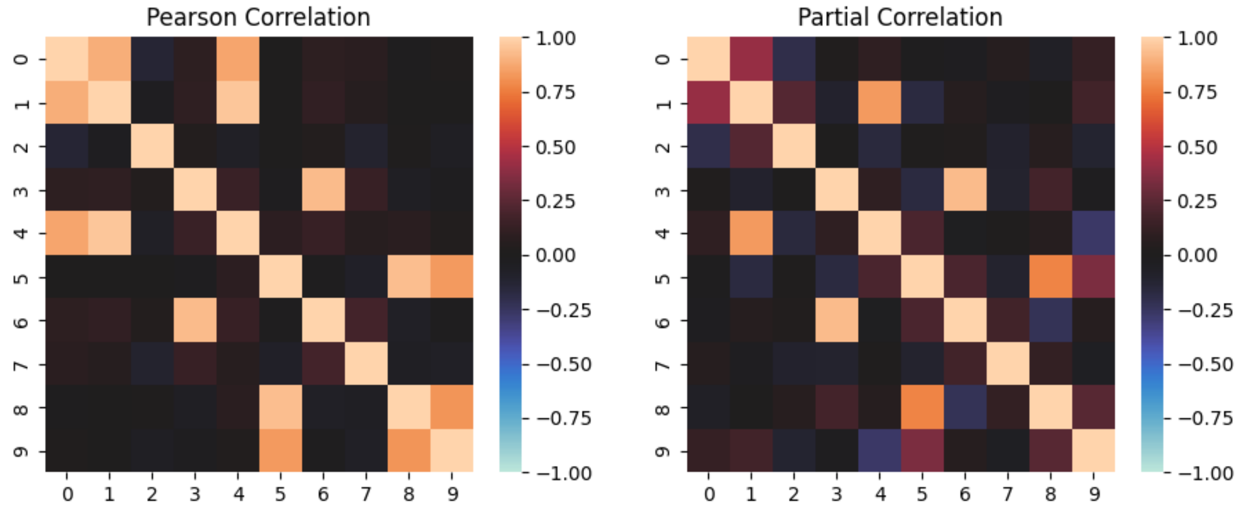


Figure 4: Pearson vs partial correlation with weights:
$[2, 10, 1, 1, 10, 1, 1, 1, 1, 1]$
$[1, 1, 1, 1, 1, 10, 1, 1, 10, 2]$
$[1, 1, 1, 10, 1, 1, 10, 1, 1, 1]$

We will be simulating a brain with 100 regions. The 100 weights for each brain are drawn from a weight function, which is created using the product of a sinusoidal function and the normal distribution.

Box 3 *Generate weights from Sinusoidal Function*

$$W(x) = amp \cdot (f_1 sin(s_1 x) + f_2 \sin(s_2 e x) + f_3 \sin(s_3 \pi x)) \cdot (2\pi)^{\left(-\frac{1}{2}\right)} \cdot e^{-\frac{1}{2}\left(\frac{(x-b)}{w}\right)^2} + 1$$

where

$W(x)$ = *weight function to draw weights from*
amp = *amplitude scale parameter for the entire function*
f_i = *amplitude scale parameter for sub-function* i
s_i = *x scale parameter for sub-function* i
w = *width scale parameter of the normal distribution*
b = *translation parameter of the normal distribution*

The parameters of the weight function can be tweaked to adjust its amplitude and frequency, as well as the proportion contributed by each sub-function. The parameters we chose were amp: 0.1, f_1: 1.9, s_1: 3, f_2: 6, s_2: 5, f_3: 4.5, s_3: 1.3, and w: 7.

The parameter b translates the normal distribution along the x-axis and is used as the measure of separability between two weight functions. Functions with values for b that are close together would have low separability, while functions with values for b that are far from each other would be easy to separate.

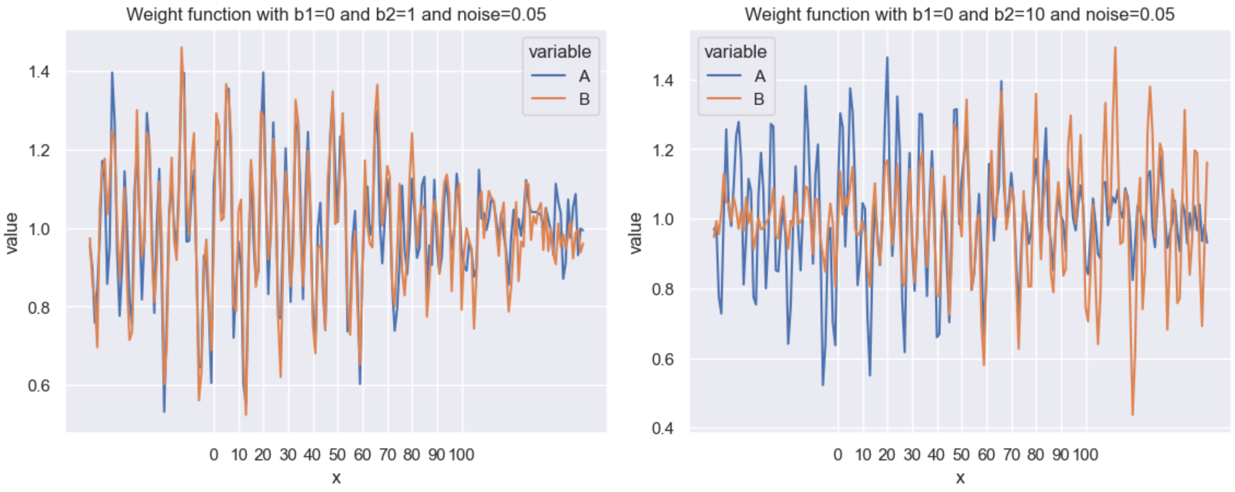


Figure 5: Comparison of weight functions at various levels of separability.

For this simulation, the b parameter of “male” brains will be situated at 0, while the b parameter of “female” brains will vary depending on the desired level of separability.

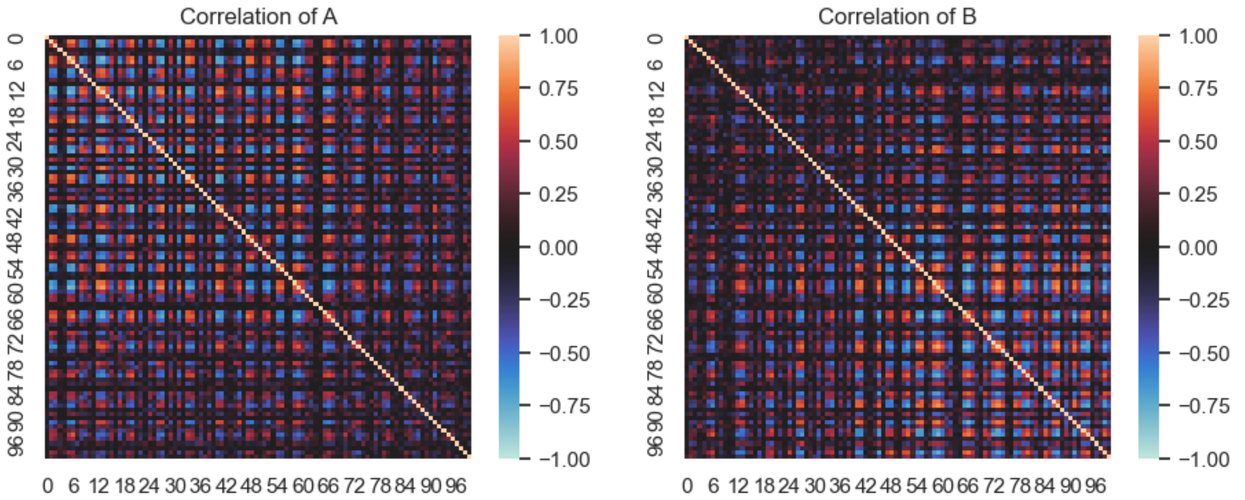


Figure 6: Pearson correlation of simulated ”male” and ”female” brains situated at $b = 0$ and $b = 10$ respectively.

In total, we generated correlation matrices for 11 levels of separability. Every level has 1000 subjects each with 100 time points and 100 brain regions. 50% of the subjects are “male” and 50% are “female”.

3.2 Exploratory Data Analysis

3.2.1 Independent Samples t -Test with Unequal Variances (Welch’s t-Test)

In statistics, an independent samples t-test is a parametric test which is used to compare the populations of two variables. The t-test can examine two types of comparisons: whether two populations have equal means, a two-sided test, or if one of the population means is greater than or equal to the other, a one-sided test.

Here, we use the two-sided independent samples t-test to test whether two populations have equal mean pairwise correlations:

- **Null Hypothesis (H_0):** Population 1 and Population 2 have the same mean.

$$\mu_1 = \mu_2$$

- **Alternative Hypothesis (H_1):** Population 1 and Population 2 *do not* have the same mean.

$$\mu_1 \neq \mu_2$$

The t-test follows three assumptions: the populations are independent, each population’s data is normally distributed, and the data within each population are independent.

We created two datasets, x_1 and x_2 by randomly sampling $j_1 = 450$ and $j_2 = 550$ pairwise correlation values at Region$_{(19,17)}$ from population 1, the 500 ”male” correlation matrices, and population 2, the 500 ”female” correlation matrices, that we previously generated in Section 3.1 under the following parameters:

amp: 0.1, f_1: 1.9, s_1: 3, f_2: 6, s_2: 5, f_3: 4.5, s_3: 1.3, w: 7 and b : 1.0.

These resulting sample distributions represent the different pairwise correlation values for $\text{Region}_{(19,17)}$ in males and females:

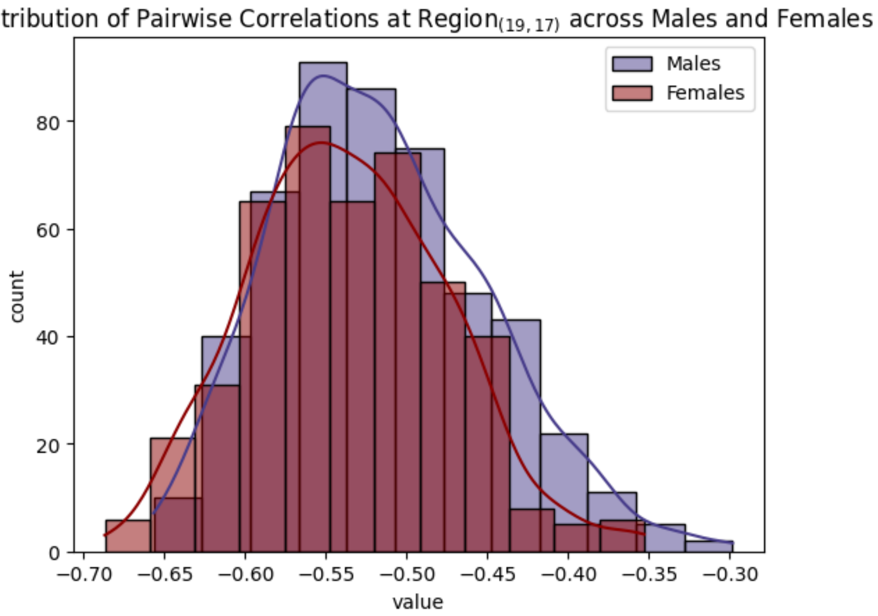


Figure 7: Distribution of Simulated Pairwise Correlations at $\text{Region}_{(19,17)}$, Sampled from Simulated Male and Female Populations at Increasing Degree of Separability

We can test the statistical significance of the difference of each sample's means by calculating the t-statistic. To calculate the t-statistic, we find the difference in sample means ($\hat{\mu}_i$) and calculate its estimated overall standard error ($\sigma_{\Delta\hat{\mu}}$), as follows:

Box 4 *t-Statistic*

$$t = \frac{\Delta\hat{\mu}}{\sigma_{\Delta\hat{\mu}}} = \frac{\hat{\mu}_1 - \hat{\mu}_2}{\sqrt{\sigma^2_{\hat{\mu}_1} + \sigma^2_{\hat{\mu}_2}}}$$

$$\sigma_{\hat{\mu}_i} = \frac{\hat{\sigma}_i}{\sqrt{j_i}}$$

where

$\hat{\mu}_1, \hat{\mu}_1$ = *sample means of population* 1 *and population* 2
$\sigma_{\hat{\mu}_i}$ = *variance of sample means of population* i
$\hat{\sigma}_i$ = *sample variance of population* i
j_i = *sample size of population* i

Using the t-distribution, the t-statistic yields a p-value that, under a given significance level

($\alpha = 0.01$), allows us to reject or fail to reject the null hypothesis.

Here, our t-statistic (≈ 4.68) yielded an approximate p-value of $0.00 < \alpha$, which means we reject the null hypothesis that our two populations have the equal mean pairwise correlations.

t-tests compare the region-wise pairwise correlations in male and female BOLD signal activity, revealing interesting insights about the overall difference in the Default Mode Network for males and females.

3.2.2 Multiple Hypothesis t-Test

When comparing the male and female brain, we examine and compare the whole pairwise correlation matrix, rather than a single pairwise correlation, as we did in Section 3.2.1. In the HCP dataset, there exists 100 regions, yielding 4950 total pairwise correlations.

We will analyze and compare each of the 4950 pairwise correlations observed in the male brain with those in the female brain. This results in a total of 4950 t-tests. This statistical testing of many variables is referred to as the multiple hypothesis or multiple comparisons independent samples t-test. We can see the null hypothesis below:

- **Null Hypothesis (H_0):** All of the population means are equal.

$$\mu_1^{female} = \mu_1^{male}, \quad \mu_2^{female} = \mu_2^{male}, \quad \mu_3^{female} = \mu_3^{male}, \quad \ldots, \quad \mu_{4950}^{female} = \mu_{4950}^{male}$$

- **Alternative Hypothesis (H_1):** At least one of the population means is not equal.

Although similar to an independent samples t-test, a multiple comparisons test yields slightly different results due to the Familywise Error Rate. We will examine the results of two datasets, one whose pairwise correlations were generated at a higher degree of separability and another whose pairwise correlations we generated at a lower degree of separability.

Using the same simulated female and male population datasets utilized in 3.2.1, we repeat the sampling process, except rather than only sampling at one pairwise region, we randomly sample at all of the pairwise regions. This results in two datasets of generic pairwise correlation matrices represented as $N = 1000$ random samples (one sample per subject); 450 of these random samples belong to the first dataset, and are drawn from the simulated population of female correlation matrices. The remaining 550 random samples belong to the second dataset, and are drawn from the simulated population of male correlation matrices. Each of these random samples have a sample size of $j = 4950$ (each subject has 4950 pairwise correlations).

Below we can see the distribution of *one* out of the 4950 simulated pairwise correlations corresponding to $\text{Region}_{(0,1)}$ (i.e. Pairwise Region 1) for males and females:

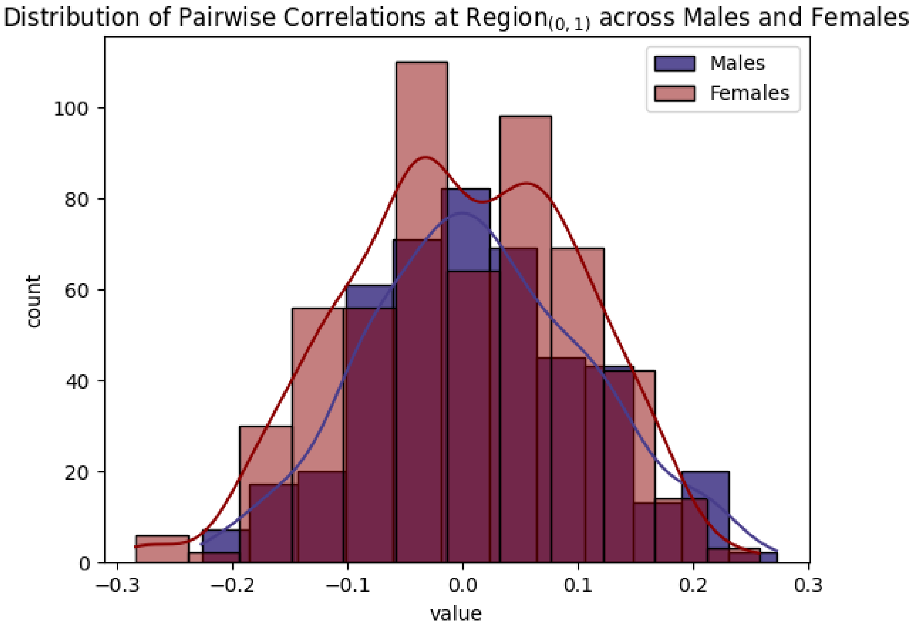


Figure 8: Distribution of Simulated Pairwise Correlations at Pairwise Region$_1$, Sampled from Simulated Male and Female Populations at Higher Degree of Separability

We will compare the difference in sample means for males and females for a total of 4950 pairwise regions. We can test the statistical significance of these differences by using the formula for the t-statistic given in Box 4.

The resultant p-values for each of the 4950 pairwise correlations is captured below:

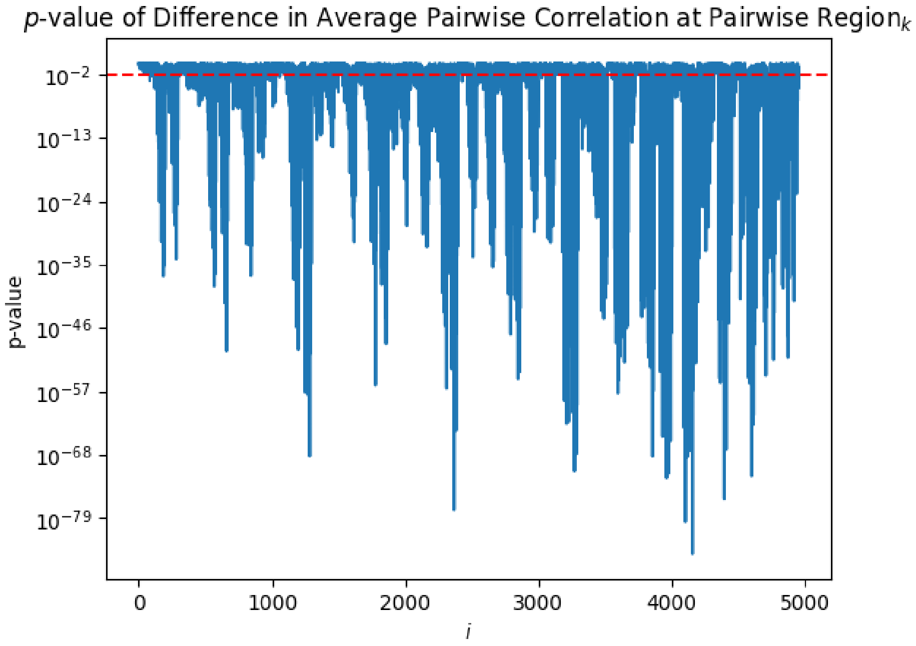


Figure 9: p-values of each Pairwise Correlation

As we can see, the t-test yielded a p-value $< (\alpha = 0.01)$ for all of the pairwise correlations, so we reject the null hypothesis. However, our significance level, $\alpha = 0.01$ is the marginal error rate. To achieve joint error rate control, we will apply the Bonferroni correction, which aids in decreasing the false positive rate when considering the real-life HCP dataset, where each of the pairwise correlations' distributions may not be as distinct between nor consistent within the males and females.

We repeat this simulation, but with two distributions that have a lower degree of separability. This will help demonstrate how the multiple comparison independent samples t-test performs on distributions with less distinct means and, thus, more sensitivity to false positives.

To generate two datasets with the same population mean, we repeat the same random sampling process, but with separability $b = 0$.

As captured in the figure below of a *single* pairwise correlation, there is less separability in the 4950 pairwise correlations between males and females:

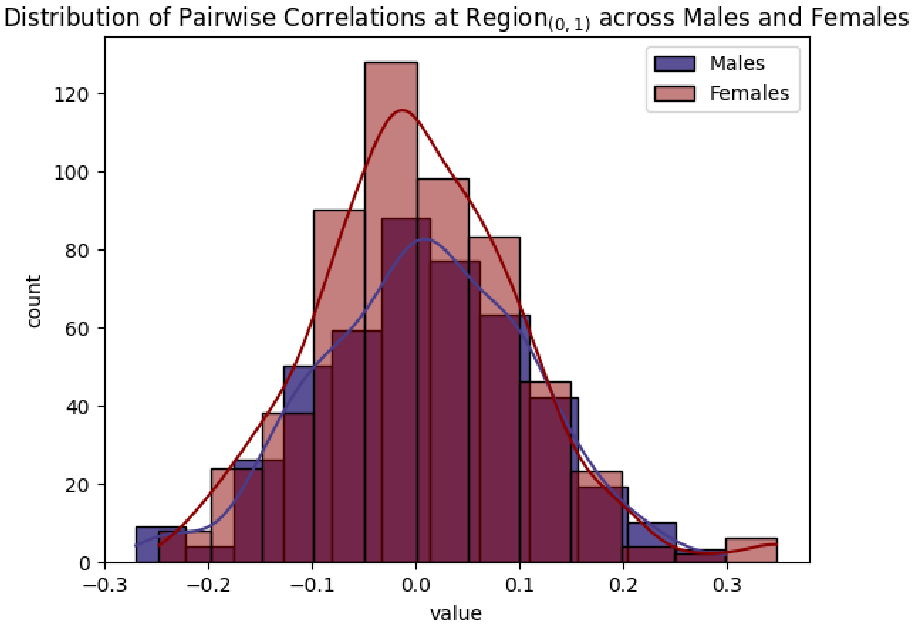


Figure 10: Distribution of Simulated Pairwise Correlations at Pairwise Region_1, Sampled from Simulated Male and Female Populations at Lower Degree of Separability

We calculated the t-statistics and their resultant p-values to observe a false positive rate greater than the marginal error rate, $\alpha = 0.01$, as captured by the high number of significant p-values below:

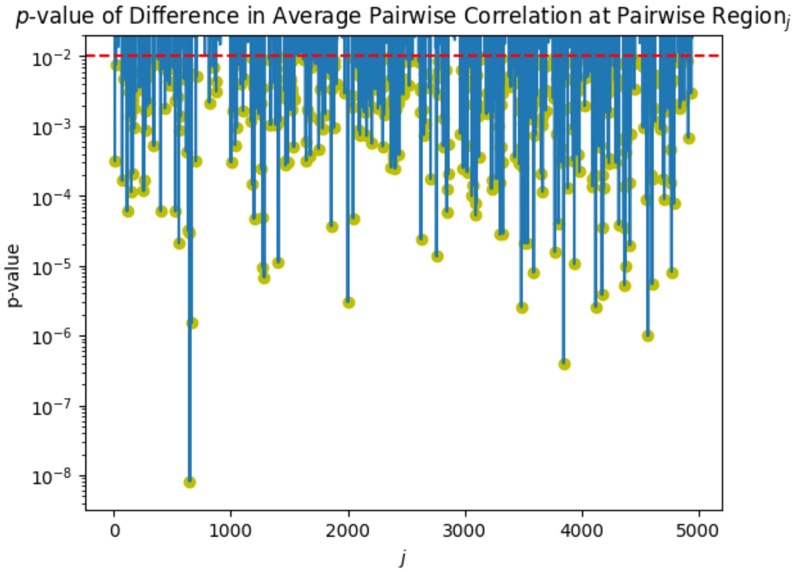


Figure 11: p-values of Each Pairwise Correlation under the Null Hypothesis

When performing any multiple comparisons two-sided independent samples t-test on variables with unknown population means, we can expect to observe a similar false positive rate. For our exploratory data analysis, we are interested in conducting a more conservative statistical test to ensure that our pairwise correlations of interest are truly different between males and females, so we apply the Bonferroni correction.

3.2.3 Bonferroni-Corrected Significance Level

When considering multiple hypothesis testing, the probability of incorrectly rejecting the null hypothesis, also referred to as the false positive rate, is equal to the significance level (α) for *each* hypothesis. The *total* false positive rate, however, is also dependent on j, or the number of hypotheses:

Box 5 *Familywise Error Rate*

$$P_{FWER} = 1 - (1 - \alpha)^j$$

where
j = the number of hypotheses

This is recognized as the Familywise Error Rate ($FWER$), or the probability of at least one false positive. Using this formula, the probability of incorrectly rejecting the null hypothesis at least once ($FWER$) during our multiple comparisons test is approximately equal to 1; incorrectly labeling a pairwise correlation as having a statistically significant difference between males and females is almost guaranteed. We can reduce this Familywise Error rate by applying the Bonferroni correction, which divides α by the number of hypotheses, or in our case, pairwise correlations $j = 4950$:

Box 6 *Bonferroni Corrected Significance Level*

$$\frac{\alpha}{j}$$

After plugging in the Bonferroni-corrected α into the Familywise Error Rate equation, assuming that j is large and α is small, our new total probability of incorrectly rejecting the null hypothesis is reduced from 1 to approximately equal to α:

Box 7 *Familywise Error Rate with Bonferroni-Correction*

$$P_{FWER} = 1 - (1 - \frac{\alpha}{j})^j \approx \alpha$$

We can apply this Bonferroni correction to the $\alpha = 0.01$ for our simulated-data multiple comparisons t-test with the lowest degree of separability, which yields the following p-values:

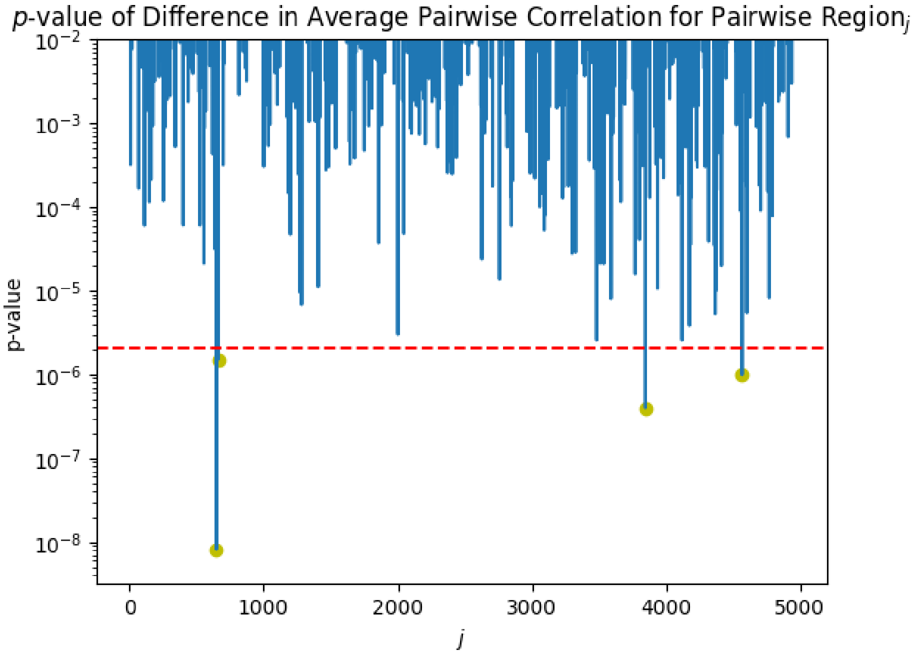


Figure 12: p-values of Each Pairwise Correlation under the Null Hypothesis with Bonferroni-Corrected Significance Level

As we can see, the prior observed false positive rate that was captured in Figure 11 reduced to approximately 0.0008 as a result of the new Familywise Error Rate ($P_{FWER} \approx (\alpha = 0.01)$).

Thus, when comparing the pairwise correlation matrices of males and females, applying the Bonferroni-correction on our significance level (α) helps yield a lower Familywise Error Rate. With a lower Familywise Error Rate, our statistical test will reveal a sparser matrix of statistically significant pairwise correlations, minimizing the number of features that will be used in our Graph Neural Network.

3.3 Modeling

3.3.1 KNN

This section will describe some of our findings and task related processes found on training the K nearest neighbors model or KNN. We decided to KNN as a baseline model for our GNN. KNN is a commonly used and powerful classifier that compares the distance between subject and data points and assigns the most common label.

For our implementation of KNN we first vectorized the correlation matrix to flatten the 4950 pairwise correlations into a single array. Then using the Sklearn package we can perform the KNN algorithm on our simulated and HCP data given the vector. In addition, the KNN algorithm we choose uses standardized distances so input data can be scaled differently compared to our training data.

3.3.2 GNN

This section will describe methods used and task related considerations that were made before making the GNN.

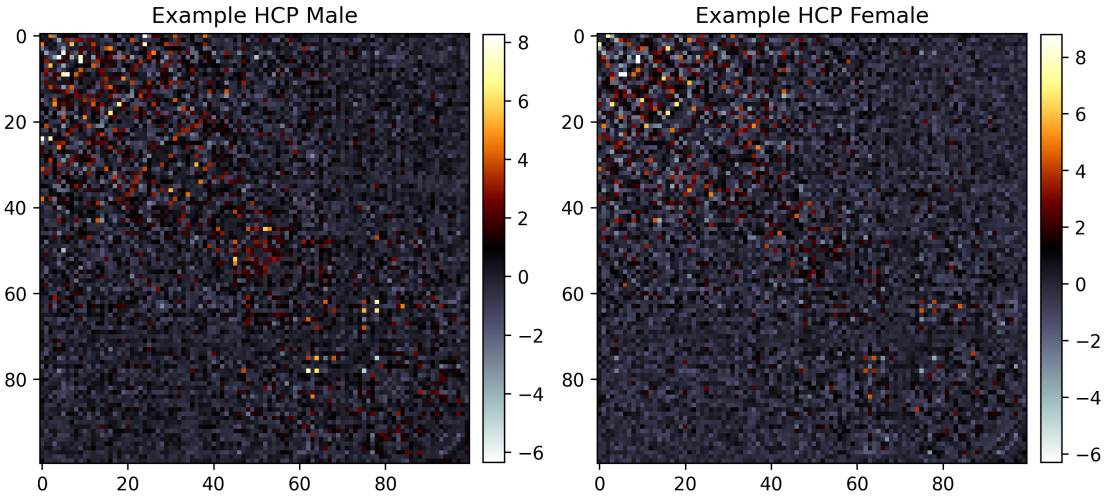


Figure 13: Example Normalized Male and Female Subjects from HCP Data

3.3.2.1 GNN input The input to our Graph Neural Network (GNN) consists of preprocessed ICA components provided by the previously discussed HCP fMRI data releases for 1,003 young adults. Each subject is associated with metadata including age range and gender. Gender information is extracted and used as a label for our classification task. Example of a male and female are in Figure 13.

We also must use thresholding to speed up the training process, which focuses the regions that have high positive or negative activity during resting state. The input required for a PyTorch Geometric GNN requires a variable called 'edge index', which determines two locations to be connected. For example, a fully connected edge index would be 4,950 connections in a 100 x 100 pairwise correlation matrix, we want to reduce this.

We took two approaches for creating the variable. The first was to extract the statistically signification correlations from the differences of the means of the male and female class, as done in the EDA section. Results leave 1,695 connections for every subject regardless if male or female when using alpha of 0.01. Using this particular method can be faster when a new edge index doesn't need to be generated for each subject. Our second approach was to threshold on individual subjects which creates an unique edge index for a subject. The method was in-part inspired by molecule classification of toxicity based on the structure of the graph (Jiang et al. 2021). This approach allows for more weights to be trained in the layers of the GNN, but different a edge index for each subject.

3.3.2.2 GNN architecture Our GNN is built on using a PyTorch Geometric (PyG) methods from which we utilized the Graph Convolutional Network (GCN) message passing technique introduced in the paper from Kipf and Welling (2016). The GCN layer is a data reduction or expansion method which takes the size of the input layer, and outputs the desired size for the output. Generally transforming the data to a higher dimension yields more complex relations from a graph. From our simulation test and EDA, the noise in the data would become a significant issue using large hidden layers. We also want to limit the number of GCN layers due to having too many would average the weights where no interesting relationships can be found.

From empirical findings of running with different hyperparameters, general rules of thumb for neural networks have been useful. This mainly includes having a tuned learning rate, using activation functions on each hidden layer, and having a linear layer before output. Some changes from usual neural networks are using the tanh activation function since our data needs to be analyzed for positive and negative correlation values. Utilizing batchnorm on each hidden layer proved to be useful, which was inspired from a paper with a similar task. Article by Hough (2022) describes the usefulness of batchnorm which is most likely due to its benefits to accelerate training per epochs. Overall there is no single architecture that will work for any given task. As tested by using same architecture on simulated fMRI data vs. real fMRI data, the results vary. Figure 14 displays an example data flow through a GCN model.

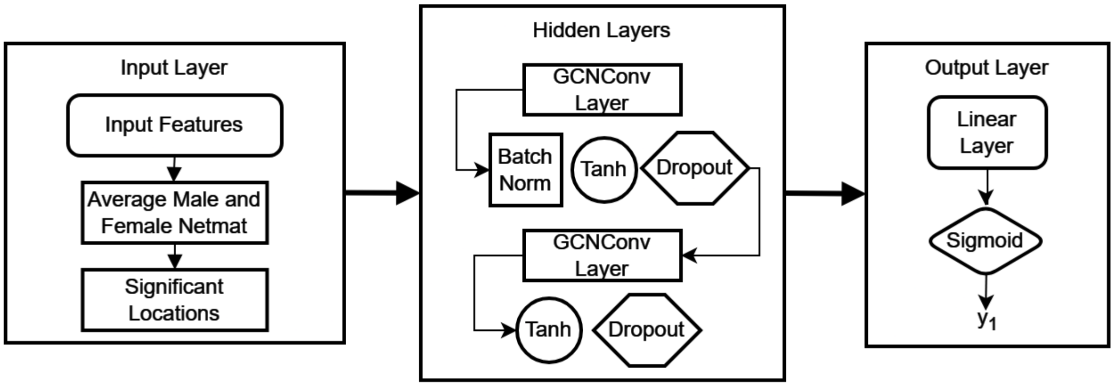


Figure 14: Example GCN Diagram

Our loss function is Binary Cross Entropy (BCE) With Log Loss, from the PyG library that combines sigmoid and BCE to have stability when calculating loss. Alternative for this would be to simply using BCE and having the GNN output the result ran through a sigmoid function. Our Optimizer function is Adaptive Moment Estimation or Adam optimizer which has become a standard in most neural networks. There are some trade-offs for each optimizer but from our experience, fine tuning the learning rate was largest difference in performance than changing the optimizer.

3.3.2.3 GNN implementation The GNN processes the graph data for each subject iteratively examining the relations of neighboring significantly connected components to a specific component in the graph. By running the network by subject, over time it will recognize important indicators in the structure found by correlation values on specific components. It summarizes these relationships and updates the state of each component per subject iteratively using GNN operations. The process is repeated until the network achieves satisfactory accuracy on both the training and validation sets. We determine satisfactory performance by finding a number of epochs, times all the data in the training subset is ran once, where the loss appears to converge. GNNs leverage parallel computing methods, making them suitable for efficient training on cloud computing platforms. Despite this, most of our training has run on memory of a personal computers.

3.3.2.4 Output of GNN Our GNN belongs to the class of graph classification GNNs, where the objective is to predict the class label of the input graph. In our case, the output is binary, indicating whether the subject is female (true) or not (false). We use a linear layer, mean across the layer, then a sigmoid if using BCE loss, otherwise the mean from the linear layer if using BCE log loss. The model's accuracy is evaluated on a test set that is held out from both training and validation. We also have access to a single UCSD's own fMRI resting state fMRI data.

4 Application

4.1 Exploratory Data Analysis: Distinctions in Functional Connectivity among Male and Female Subjects

4.1.1 Average Functional Connectivity among Male and Female Subjects

We are interested in whether there is a difference in male and female fMRI resting state data. In order to test this difference, we compare the pairwise correlation matrices of roughly 500 male subject and 500 female subjects, as provided in the HCP dataset. We aggregated the male subjects' and the female subjects' pairwise correlation matrices into a single matrix that contains the average correlation for each of the unique pairwise regions:

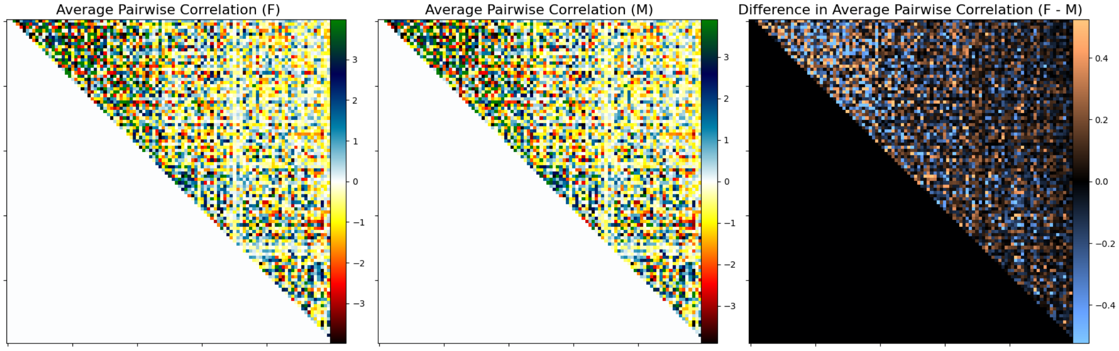


Figure 15: Average Functional Connectivity of Male and Female Subjects from HCP Data as Pairwise Correlations

To further explore the possible differences in male and female fMRI resting state data, we represented the averaged correlation matrices for each sex as a histogram and found that, to the naked eye, there are no visually obvious differences between male and female fMRI resting state data:

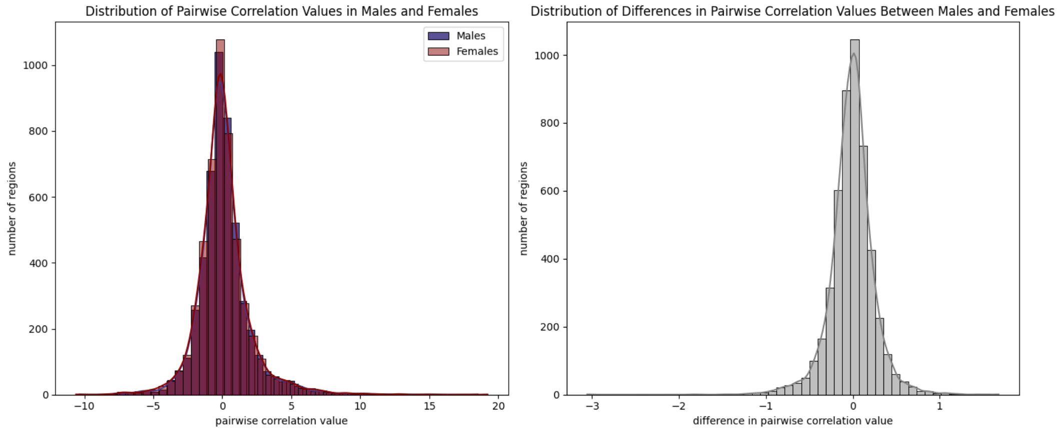


Figure 16: Distribution of Average Functional Connectivity of Male and Female Subjects from HCP Data

However, we can conduct a more formal test that better quantifies whether the observed difference between male and female data is statistically significant.

4.1.2 Conducting a Two-Sided Independent Samples t-Test to Compare Average Functional Connectivity in Males and Females

We begin by checking the necessary assumptions:

1. Populations are independent of one another.
   - We know that male pairwise correlation matrices are independent from female pairwise correlation matrices.
2. Populations are normally distributed.
   - The Fischer Transform of correlation values produces Gaussian distributions for each pairwise region across all subjects in the respective sex.
3. Data within each population are independent.
   - We know that the correlation matrices of same-sex subjects are independent from each other.

Then, we construct our hypotheses:

- **Null Hypothesis (H_0):** All of the population means of pairwise correlations are equal among men and women.

$$\mu_1^{female} = \mu_1^{male}, \quad \mu_2^{female} = \mu_2^{male}, \quad \mu_3^{female} = \mu_3^{male}, \quad \ldots, \quad \mu_{4950}^{female} = \mu_{4950}^{male}$$

- **Alternative Hypothesis (H_1):** At least one of the population means is not equal.

After conducting the t-test, we found that of all unique pairwise regions, 427 ($\approx$ 8.63%) have statistically significant differences between males and females. The significant regions are visualized below:

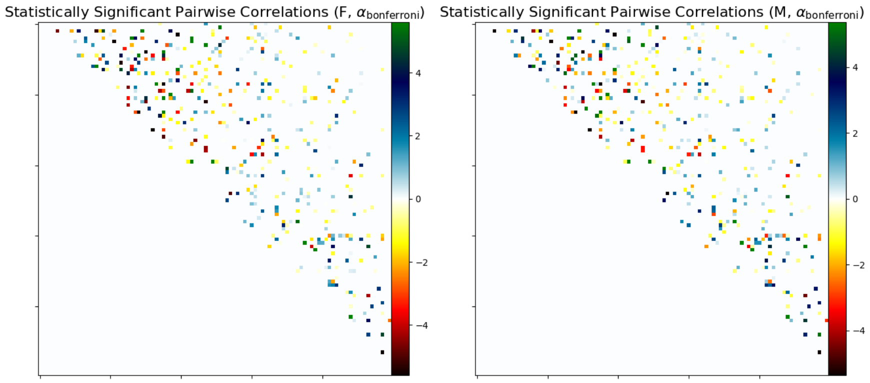


Figure 17: Comparing Male and Female Pairwise Correlations with Statistically Significant Differences

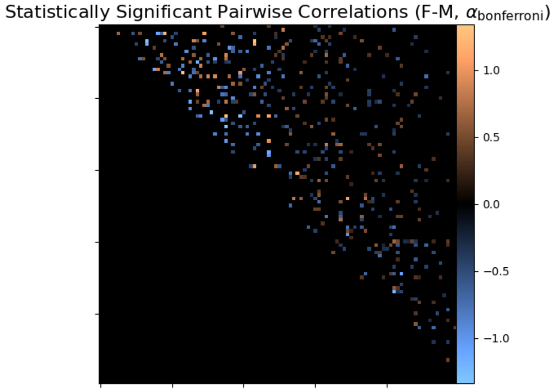


Figure 18: Pairwise Regions with Statistically Significant Differences in Correlation between Males and Females

As we can see, the statistically significant pairwise regions that have high differences, as indicated by their bright colors, are clustered on the correlation matrix. In comparing the magnitudes of all differences, we also found that female subjects displayed stronger pairwise correlation values, which is in accordance with findings by Sie et al. (2019).

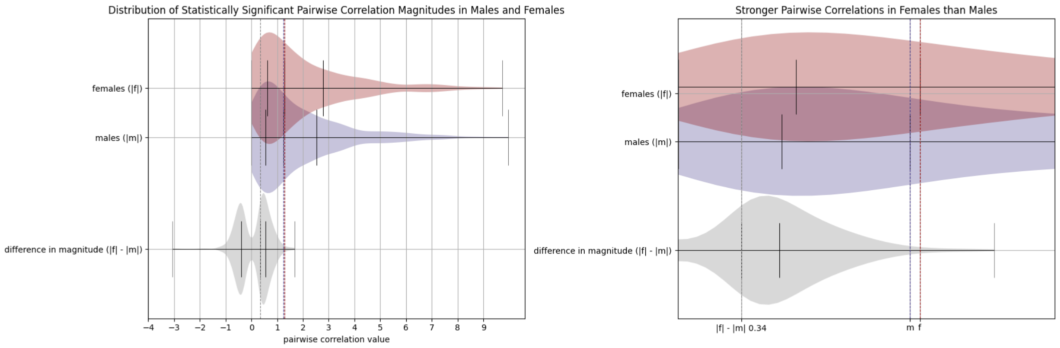


Figure 19: Comparing Male and Female Significant Pairwise Correlation Magnitudes

Standard error was estimated via bootstrapping and were estimated as the standard deviation in sample means of simulated medians. The values for the female median, male median and difference in female versus male medians are as follows (respectively): ≈ 0.00618, ≈ 0.00427, ≈ 0.00141.

The HCP data's estimates of functional connectivity between males and females does not suggest significant differences between all pairwise regions, but some do have significant p-values. Of the significant pairwise regions, the female pairwise correlations are observed to have slightly higher magnitudes. These significant regions support the success of our classification task.

4.2 GNN Description

The models in their structure are similar but some parameters are set different. The architecture is similar to the one described in the Modeling section. Figure 20 describes each step that is ran in both of the models, the image is ran twice since we found using two layers is enough. The following will describe the best performing parameter combinations for each GCN model we produced.

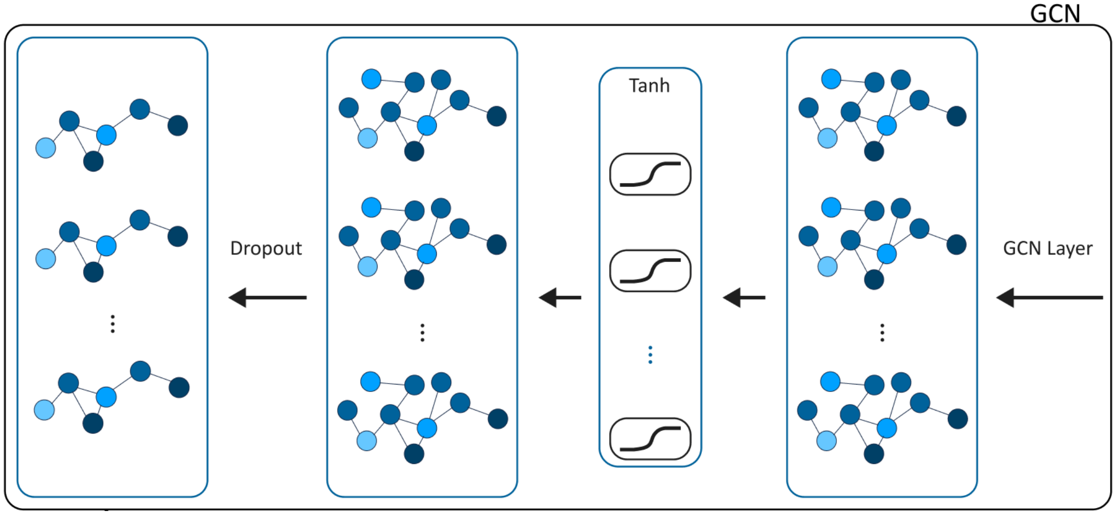


Figure 20: Architecture Used For Both GCN Models

4.2.1 GCN Model: Edge Threshold

To begin the data is normalized, then using the edge threshold method to generate our input edge index, we result in different shaped graphs as an input to the model. Figure 21 shows an example of the network structure when using the threshold of 0.75 on the correlation matrix. Also noting the result is a connected graph but not a complete graph.

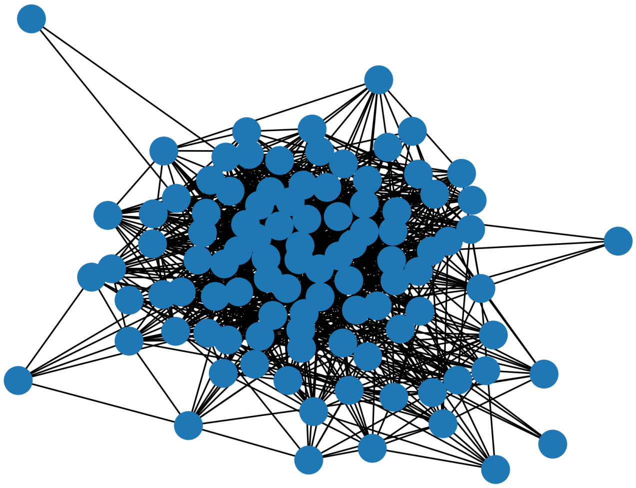
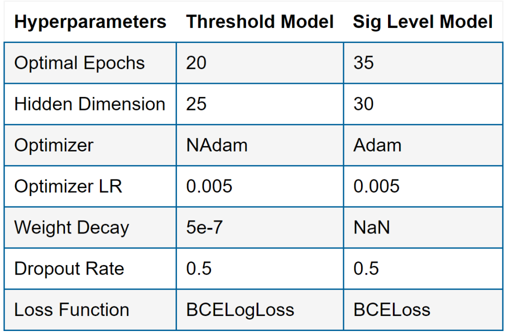


Figure 21: Example Edge Threshold = 0.75 Network

The hyperparameters are tuned to yield the best and stable results when the code is ran again. Figure 22 summarizes the findings for optimal hyperparamaters for each model. Optimal epochs refers to the number of epochs that should be ran to achieve high accuracy. Hidden dimensions describes the models dimension reduction from a 100.

Outputs of the model were evaluated based confusion tables to optimize even false positive and false negative chances due to the task having equal penalty for both wrong guesses.



Hyperparameters	Threshold Model	Sig Level Model
Optimal Epochs	20	35
Hidden Dimension	25	30
Optimizer	NAdam	Adam
Optimizer LR	0.005	0.005
Weight Decay	5e-7	NaN
Dropout Rate	0.5	0.5
Loss Function	BCELogLoss	BCELoss

Figure 22: Hyperparameter Summary Table

4.2.2 GCN Model: Significant Level Threshold

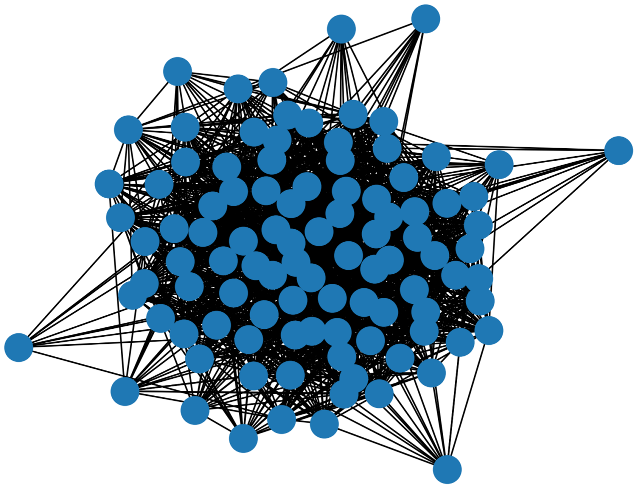


Figure 23: Significant Level = 0.1 Threshold Network

The significant level thresholding model was similar the simple thresholding method, where one of the main differences is the input edge index. The edge index is identical for each subject, so it depends on the message passing in training how the weights are trained. Where as, the simple thresholding method weights are updated in different regions but with apply to graphs with similar structure.

5 Results

5.1 Simulation Comparison

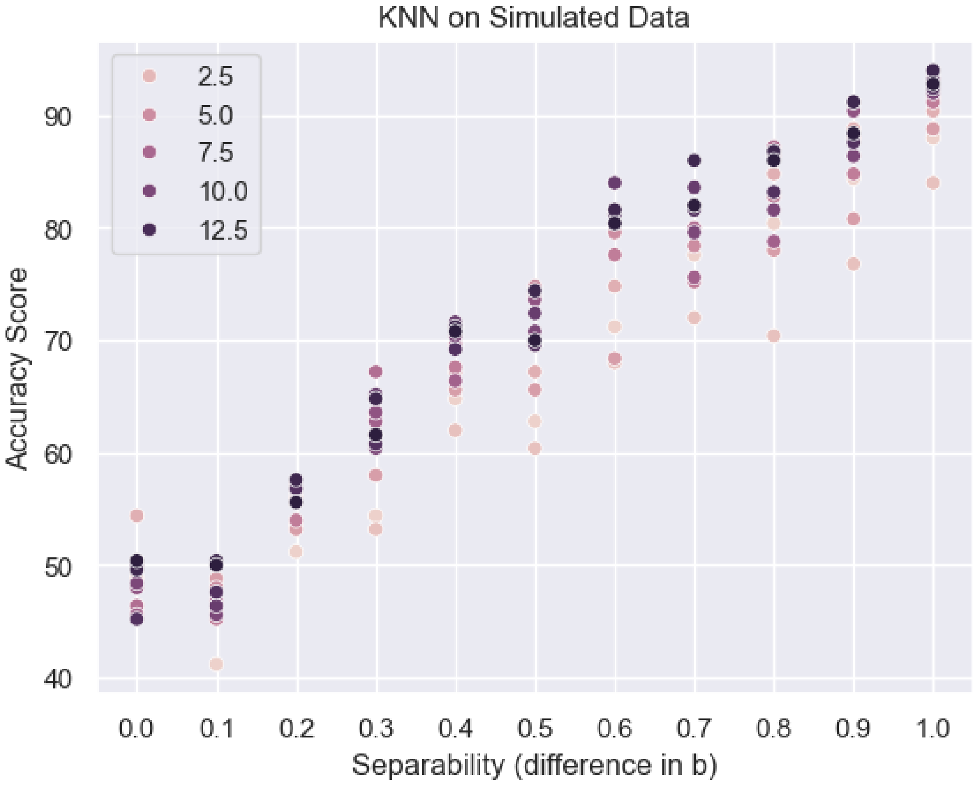


Figure 24: K-NN accuracy at 11 levels of separability, with k being indicated by hue.

We used the simulated correlation matrices to test our classification methods at varying levels of separability. From the graph above, it is clear that the separation accuracy of K-NN increases with a greater difference in the b parameter. At higher levels of separability (difference in b above 1), K-NN is able to separate the two classes with 100% accuracy.

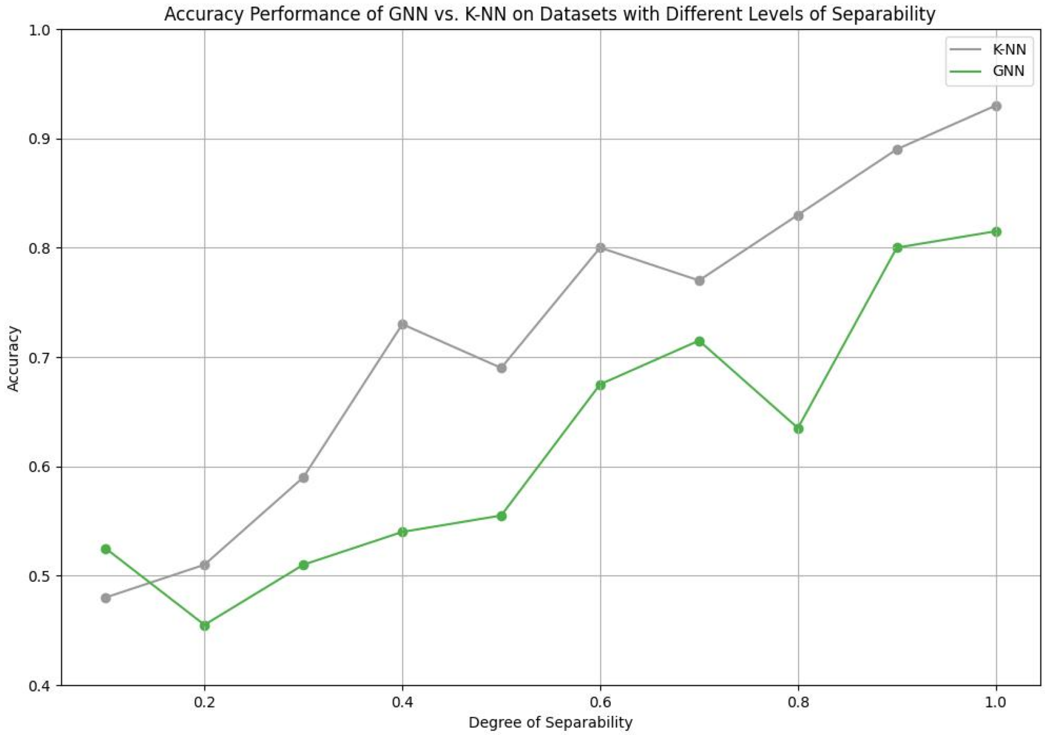


Figure 25: GNN vs K-NN at varying degrees of separability

There is a positive relationship between model accuracy and separability of groups for both Graph Neural Networks and K-Nearest Neighbours. The K-NN model outperforms the GNN model at the same degree of separability. This is likely due to the relatively small number of subjects, which can be helpful for K-NN models but is a limiting factor for Graph Neural Networks. This limitation exists in the HCP dataset as well.

5.2 Model Results

Figure 26 and Figure 27 summarize our results where the models are given 2 standard deviations of confidence of accuracy results. These results for both GCN models have higher variance on accuracy due to randomness of each time the models are ran. Best performing on average is the Edge Threshold GCN model, but the most consistent model is using k-NN.

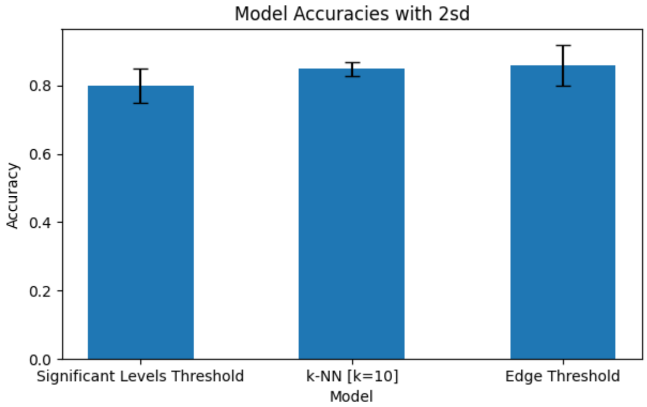
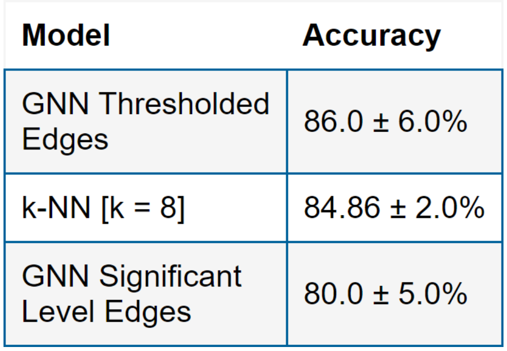


Figure 26: Summary Bar plot of Model Accuracies



Model	Accuracy
GNN Thresholded Edges	86.0 ± 6.0%
k-NN [k = 8]	84.86 ± 2.0%
GNN Significant Level Edges	80.0 ± 5.0%

Figure 27: Results Accuracy Table

After we were satisfied with the performance of all the models, we tested on our UCSD single subject resting state fMRI. The models all predicted the correct label, but having only a single test point from a single gender is a bit of a limitation, but still an interesting test case for testing the model outside of the HCP dataset.

6 Discussion

6.1 Data Limitations

First off, the dataset that was provide by HCP was not significantly large enough for complex classification models. With 1000 data points our standard errors were around 3.5 percent since we were only able to use 100 points for our accuracy tests. This problem is difficult to avoid due to the scarcity and cost of fMRI brain data as each scan takes expensive resources and lots of time.

We attribute this limitation to the result of our KNN model and GNN model having no significant difference in accuracy. Given the amount of data it would be easy to assume that KNN would perform better since it works well with limited amounts of data versus a GNN that needs lots of data to train its hyper parameters. As the amount of fMRI data becomes more and more accessible we assume that the GNN will out perform the KNN not only in accuracy but computationally in speed. As the amount the of data points increases the KNN algorithm becomes slower whereas the GNN becomes more accurate.

In addition to data shortages, there is also a limitation in data formatting. Each type of fMRI scan needs to go through a preprocessing pipeline to make the data clean and usable for analysis. However, there are many different types of data preprocessing methods commonly used, making it difficult to compare fMRI datasets.

For example, the HCP data has on average higher correlation values compared to the UCSD data gathered from Andrew's brain scan even after applying the same transformations. This can be due to a higher sensitivity of the HCP MRI machines or any various factors. As such, it's important to use models that have standardize inputs for the future.

6.2 Methodology Limitations

6.2.1 Autocorrelation and Estimating Functional Connectivity

Functional connectivity is estimated using Pearson's Correlation, which first standardizes the given time series of BOLD signal activity per voxel. This standardization process utilizes the Z-scores of these data points, however, the autocorrelation present in the time series inflates these Z-scores. This effect of autocorrelation may have impacted the results of our Graph Neural Network due to its influence on the dimensionality reduction processes that occurs within the Graph Convolutional Network Layers, given that it relies on the aggregation of our Fisher-transformed Pearson's Correlation values to map the input to the final feature space. Due to lack of time and resources, we did not correct for autocorrelation before feeding in the matrices to our Graph Neural Network.

6.2.2 Joint Error Rate Control: Significant Level Threshold

When selecting the significance level threshold during statistical testing, there is an inherent trade-off between false positive rate and false negative rate. Applying the Bonferroni correction to our alpha significantly increases the false negative rate, possibly ”disqualifying” significant pairwise correlations that would have otherwise contributed to identifying differences in the functional connectivity of males and females. This disqualification may have influenced the Graph Neural Network by excluding important pairwise regions from the training and validation datasets, limiting its ability to extract valuable patterns when performing the prediction task. Alternatively, however, having a high false positive rate may retain noise that could result in poor performance due to overfitting. Ultimately, due to lack of time and resources, we did not employ any selection processes to tune for the ideal significance level (in terms of yielding a higher-accuracy model) after observing that the model still performed moderately well.

6.2.3 Input Feature Selection: Edge Threshold

The edge threshold used to prune the edges of our input features (i.e. the graph representation of each subject’s functional connectivity) had a significant impact in model performance. While testing the Graph Neural Network, we tuned this hyperparameter manually, evaluating the model training and validation accuracies in conjunction with the training loss, increasing the edge threshold accordingly to maximize the performance of our prediction-task. The Graph Neural Network performed best when pruned using edge thresholds that removed brain regions with Fisher-transformed pairwise correlations less than (approximately) 0.1 or 1. Due to the infinite number of possible thresholds, and lack of resources, our Graph Neural Network’s performance may have been limited to the effectiveness of our final threshold.

References

Hough, Sidney. 2022. "GNNS in neuroscience: Graph convolutional networks for fmri analysis." *Medium*. [Link]

Jiang, Dejun, Zhenxing Wu, Chang-Yu Hsieh, Guangyong Chen, Ben Liao, Zhe Wang, Chao Shen, Dongsheng Cao, Jian Wu, and Tingjun Hou. 2021. "Could graph neural networks learn better molecular representation for drug discovery? A comparison study of descriptor-based and graph-based models." *Journal of Cheminformatics* 13 (1). [Link]

Kipf, Thomas N, and Max Welling. 2016. "Semi-Supervised Classification with Graph Convolutional Networks." *arXiv preprint arXiv:1609.02907*. [Link]

Sie, Jia-Hong, Yin-Hua Chen, Yuo-Hsien Shiau, and Woei-Chyn Chu. 2019. "Gender- and Age-Specific Differences in Resting-State Functional Connectivity of the Central Autonomic Network in Adulthood." *Front Hum Neurosci* 13 , p. 369. [Link]

Acknowledgements

We want to extend our thanks to our mentor, Professor Armin Schwartzman, who graciously provided his extensive domain expertise in statistics, neuroscience, signal processing and machine learning throughout the project timeline.

We also want to express our gratitude to our PhD advisor, Gabriel Riegner, for his thoughtfulness and constant support.

Data were provided [in part] by the Human Connectome Project, WU-Minn Consortium (Principal Investigators: David Van Essen and Kamil Ugurbil; 1U54MH091657) funded by the 16 NIH Institutes and Centers that support the NIH Blueprint for Neuroscience Research; and by the McDonnell Center for Systems Neuroscience at Washington University.