GNN Pipeline Test Plan

Description of Overall Test Plan

This project applies graph neural networks for the task of fingerprinting functional connectomes. The project code consists of a deep learning pipeline that generates the FC data, then trains and evaluates the chosen model. There are two major pieces to be tested: the pipeline used to train and evaluate these models, and the models themselves. Testing the pipeline involves testing each of its individual parts: FC generation, dataset construction, model creation, model training, model evaluation, and the pipeline as a whole. Testing the models themselves involves testing the correctness of individual components, such as propagation or convolutional layers, and the tensor operations used in those components. Model performance is also tested on randomly generated data to ensure the model behaves as expected.

Test Case Descriptions

D1.1 – Identifier	Dataset Integrity
D1.2 – Purpose	Ensure that there is no leakage of data between train/validation/test sets
D1.3 – Description	IDs in the format of (subject, FC) are processed alongside the FC pairs. These IDs are checked after pair creation and dataset split to ensure that there are no identical pairs occurring in the training, validation, and test sets. An error is thrown if a duplicate pair is found. A successful test prints to the console.
D1.4 – Inputs	training, validation, and test set IDs
D1.5 – Expected Output	No error thrown and test success indicated by the console
D1.7 – Normal/Abnormal/Boundary	Normal
D1.8 – Blackbox/Whitebox	Whitebox
D1.9 – Functional/Performance	Functional
D1.10 – Unit/Integration	Unit
D1.11 – Results	Passed; no error was thrown and test success was indicated in console

D2.1 – Identifier	Dataset Distribution
D2.2 – Purpose	Ensure subjects, FCs are evenly distributed in pairs
D2.3 – Description	The dataset IDs created when the dataset is built are plotted and visually examined to ensure each is sampled in a uniform distribution with little to no variance.
D2.4 – Inputs	IDs corresponding to the subjects and FCs used in each pair
D2.5 – Expected Output	A plot showing a uniform distribution
D2.7 – Normal/Abnormal/Boundary	Normal
D2.8 – Blackbox/Whitebox	Whitebox
D2.9 – Functional/Performance	Functional
D2.10 – Unit/Integration	Unit

D2.11 – Results	The generated plot shows the expected uniform distribution of IDs

M1.1 – Identifier	MLP Embedding Model
M1.2 – Purpose	Test the functionality of the MLP that embeds graphs.
M1.3 – Description	One of the models evaluated in this project is a MLP that embeds a graph. To test the model, it is trained and evaluated on a random FC dataset.
M1.4 – Inputs	At the highest level, the input is a configuration specifying that the embedding MLP should be trained on a random FC dataset. Within the pipeline, the MLP model and the dataset are used as inputs in the training and evaluation functions.
M1.5 – Expected Output	The MLP should converge on the training dataset, but validation accuracy should cap at around 50% since the FCs are randomly generated.
M1.7 – Normal/Abnormal/Boundary	Normal
M1.8 – Blackbox/Whitebox	Whitebox
M1.9 – Functional/Performance	Functional
M1.10 – Unit/Integration	Unit
M1.11 – Results	The MLP performed as expected on the random FC dataset.

M2.1 – Identifier	GMN Model
M2.2 – Purpose	Ensure the correctness of the functions and tensor operations used in the graph matching network.
M2.3 – Description	The GMN takes a pair of graphs and returns a pair of embeddings for each. The operations of the GMN must individually be checked for correctness. A GMN with low dimensionality is used to process a pair of small test graphs. The output of each operation is then checked on this pair of test graphs for correctness, which is possible due to the simplicity of the network in this case.
M2.4 – Inputs	A pair of graphs
M2.5 – Expected Output	The output of each operation in the GMN and the final embeddings for each graph.
M2.7 – Normal/Abnormal/Boundary	Normal
M2.8 – Blackbox/Whitebox	Whitebox
M2.9 – Functional/Performance	Functional
M2.10 – Unit/Integration	Unit
M2.11 – Results	The outputs of each operation were checked for correctness against the expected output of each.

M3.1 – Identifier	S-GCN Model
M3.2 – Purpose	Ensure the correctness of the functions and tensor operations used in the Siamese graph convolutional network.
M3.3 – Description	The tensor operations in the Siamese graph convolutional network are examined using a pair of simple test graphs. The GCNs in this test use just two low-dimensional spectral filter layers. The outcomes of the operations are compared to their known correct outputs.
M3.4 – Inputs	A pair of simple test graphs
M3.5 – Expected Output	The correct outputs corresponding to each tensor operation in the Siamese GCN
M3.7 – Normal/Abnormal/Boundary	Normal
M3.8 – Blackbox/Whitebox	Whitebox
M3.9 – Functional/Performance	Functional
M3.10 – Unit/Integration	Unit
M3.11 – Results	The outputs of the tensor operations in the S-GCN matched their expected outputs for the pair of test graphs.

G1.1 – Identifier	Random Walk Correctness
G1.2 – Purpose	Ensure random walks behave as expected
G1.3 – Description	A random walk starts at a node and moves to a random neighbor. The random walk is run on a 10x10 matrix that is not fully connected, and the paths generated by the random walks are visually examined for correctness. Additionally, the case where a node has no edges throws an error as it should not happen.
G1.4 – Inputs	A 10x10 test matrix with only 10 edges.
G1.5 – Expected Output	Random walk paths that correspond to random neighbor selection
G1.7 – Normal/Abnormal/Boundary	Normal
G1.8 – Blackbox/Whitebox	Blackbox
G1.9 – Functional/Performance	Functional
G1.10 – Unit/Integration	Unit
G1.11 – Results	The paths generated by the random walks exhibited the expected qualities and had no irregularities.

G2.1 – Identifier	Co-occurrence Frequency Matrix Correctness
G2.2 – Purpose	Ensure that the co-occurrence frequency matrix generated by the random walks is correct
G2.3 – Description	The function generating the frequency matrix is applied to a test graph, and its output is examined for irregularity.
G2.4 – Inputs	A 10x10 test matrix
G2.5 – Expected Output	A frequency matrix highlighting the community structure of the test matrix. Elements corresponding to no edge in the test matrix should correspond to a value of 0 in the co-occurrence

	frequency matrix. Additionally, groups of nodes that are more interconnected should have higher values in the co-occurrence frequency matrix.
G2.7 – Normal/Abnormal/Boundary	Normal
G2.8 – Blackbox/Whitebox	Whitebox
G2.9 – Functional/Performance	Functional
G2.10 – Unit/Integration	Unit
G2.11 – Results	The co-occurrence frequency matrix exhibits the expected qualities

G3.1 – Identifier	Node/Feature Extraction
G3.2 – Purpose	Ensure the correctness of the code that gets the tensors of node features, edge features, and vertices list from a given list of graphs
G3.3 – Description	The inputs to the GMNs are lists of node features, edge features, and edge vertices, which must be obtained from a given list of graphs. This functionality is critical and must be checked for correctness by enumerating the lists of features and vertices for a set of test graphs and comparing it to the output.
G3.4 – Inputs	A list of test graphs
G3.5 – Expected Output	Lists of node features, test features, and edge vertices as tensors
G3.7 – Normal/Abnormal/Boundary	Normal
G3.8 – Blackbox/Whitebox	Whitebox
G3.9 – Functional/Performance	Functional
G3.10 – Unit/Integration	Unit
G3.11 – Results	The edge features corresponding to each pair of edge vertices were correct for the given list of graphs.

L1.1 – Identifier	Loss Function Correctness
L1.2 – Purpose	Test the correctness of the loss functions
L1.3 – Description	The loss for five test similarity values and five test embeddings are computed by hand using the loss function and compared to the output from the Python loss functions to ensure their correctness.
L1.4 – Inputs	A similarity value for the similarity loss function, or a pair of embeddings for the embedding loss function
L1.5 – Expected Output	The same value as that obtained from manually calculating the loss for the same similarity value or pair of embeddings
L1.7 – Normal/Abnormal/Boundary	Normal
L1.8 – Blackbox/Whitebox	Whitebox
L1.9 – Functional/Performance	Functional
L1.10 – Unit/Integration	Unit
L1.11 – Results	The outputs of the loss functions matched those manually calculated for the same inputs.

A1.1 – Identifier	Accuracy Function Correctness
A1.2 – Purpose	Test the correctness of the accuracy functions
A1.3 – Description	The accuracy for five test similarity values and five pairs of test embeddings are computed by hand using the accuracy functions and compared to the output from the Python accuracy functions to ensure their correctness.
A1.4 – Inputs	A similarity value for the similarity accuracy function, or a pair of embeddings for the embedding accuracy function
A1.5 – Expected Output	The same value as that obtained from manually calculating the accuracy for the same similarity value or pair of embeddings
A1.7 – Normal/Abnormal/Boundary	Normal
A1.8 – Blackbox/Whitebox	Whitebox
A1.9 – Functional/Performance	Functional
A1.10 – Unit/Integration	Unit
A1.11 – Results	The outputs of the accuracy functions matched those manually calculated using the same inputs.

I1.1 – Identifier	Training and Evaluation Integration
I1.2 – Purpose	Ensure that the model training/evaluation loop correctly integrates its pieces by training a simple MLP on a test dataset.
I1.3 – Description	A simple MLP is trained on the banknote authentication dataset, a binary classification task involving 5 features. The model should quickly learn to distinguish the positive and negative classes with smooth loss and accuracy curves, as this task is standard and simple. Additionally, while performance is not the main focus of this test, the training should take very little time due to the simplicity of the test model and dataset. If training can be measured in minutes with this problem, then there must be issues impacting the performance of the training pipeline.
I1.4 – Inputs	Configuration specifying the test dataset and MLP are to be used
I1.5 – Expected Output	The model quickly converges with no irregularities in the loss curve.
I1.7 – Normal/Abnormal/Boundary	Normal
I1.8 – Blackbox/Whitebox	Whitebox
I1.9 – Functional/Performance	Functional
I1.10 – Unit/Integration	Integration
I1.11 – Results	The model converged within 10 epochs using the banknote authentication data, demonstrating that the training/evaluation loop has no major flaws.

I2.1 – Identifier	Random Dataset Integration
I2.2 – Purpose	Ensure random data generation works with models
I2.3 – Description	There are two kinds of random FC datasets that can be used in the pipeline, one where random FCs are generated for 1000 “subjects” and then sampled for pairs, and another where each pair in the dataset uses newly generated FCs. Both must be tested with training and evaluation on the various models to ensure the system works.
I2.4 – Inputs	Configurations indicating random FC dataset should be used for training and evaluation
I2.5 – Expected Output	Training accuracy increases while validation accuracy stays at around 50% throughout training. Additionally, there should be no errors or irregularities withing the results directory after the pipeline is finished.
I2.7 – Normal/Abnormal/Boundary	Normal
I2.8 – Blackbox/Whitebox	Whitebox
I2.9 – Functional/Performance	Functional
I2.10 – Unit/Integration	Integration
I2.11 – Results	Model performed as expected, with training set accuracy increasing while validation did not, and the results directories were populated with the appropriate plots

I3.1 – Identifier	Pipeline Integration
I3.2 – Purpose	Ensure pipeline works from start to finish for both cross-validation and multiple-run executions
I3.3 – Description	The pipeline is run on each model using both multiple runs and cross-validation for a short number of epochs. The contents of the results directory are checked against the expected contents. This integration ensures that each piece of the pipeline works when used together.
I3.4 – Inputs	Configuration dictionary in main.py
I3.5 – Expected Output	The models are trained and evaluated with the most recent training and evaluation metrics printed to the console at every epoch. Additionally, after each execution, a full results directory containing the saved classifications and accuracy, AUC, confusion, gradient norm, loss, and margin plots.
I3.7 – Normal/Abnormal/Boundary	Normal
I3.8 – Blackbox/Whitebox	Whitebox
I3.9 – Functional/Performance	Functional
I3.10 – Unit/Integration	Integration
I3.11 – Results	Each results directory was populated with the expected plots

Test Case Matrix

Identifier	Normal, Abnormal, Boundary	Blackbox, Whitebox	Functional, Performance	Unit, Integration
D1	Normal	Whitebox	Functional	Unit
D2	Normal	Whitebox	Functional	Unit
G1	Normal	Whitebox	Functional	Unit
G2	Normal	Whitebox	Functional	Unit
G3	Normal	Whitebox	Functional	Unit
M1	Normal	Whitebox	Functional	Unit
M2	Normal	Whitebox	Functional	Unit
M3	Normal	Whitebox	Functional	Unit
L1	Normal	Whitebox	Functional	Unit
A1	Normal	Whitebox	Functional	Unit
I1	Normal	Whitebox	Functional	Integration
I2	Normal	Whitebox	Functional	Integration
I3	Normal	Whitebox	Functional	Integration