Codes and Data Availability

Data Availability

The data is collected in-house and due to proprietary reasons, the data cannot be released.

Codes

The following is a short description of all the codes that are used and uploaded with the submission.

Our codes contain 6 different folders. Here, we describe each of these folders along with some description about the files within each folder.

Preprocessing

The codes within this folder have been used to generate the data which is used in repeated 10-fold cross validation and during pretraining.

create_train_test_gcn_joint_mode.m: This matlab code is used to generate the training and validation data which is used during pretraining.

create_multiple_runs.m: This matlab code is used to generate the training, validation, and test split of out imaging genetics data which are used for repeated 10-fold cross validations.

cv_partiton.m: This code is used to generate train and validation indices.

generate_training_nback.m: Here we generate the contrast maps of our training Nback data which are used as imaging inputs to our model.

generate_training_sdmt.m: Here we generate the contrast maps of our training SDMT data which are used as imaging inputs to our model.

generate_testing_nback.m: This generates the contrast maps of out testing Nback data.

generate_testing_sdmt.m: This generates the contrast maps of out testing SDMT data.

Pretraining

The codes within this folder are used to pretrain the genetic branch and the classifier of GUIDE.

wrapper.py: This python code is used to train the genetic branch and the classifier of our model. We save the performance for each epoch which are later used.

find_checkpoints.m: Here we track the validation performance to find the early stopping point.

model_random.py: The model architecture of the genetic branch and classifier of GUIDE.

net_train.py, net_test.py: These scripts used to train and test our model.

Cross Validation

The codes within this folder are used to cross validate GUIDE. The performance obtained here is shown in Table. 1.

wrapper_img.py: We warm start out model and train it on the imaging-genetics data in repeated 10-fold validation setting. This script trains the model, saves the intermediate results, plot them, and also store the model parameters as checkpoints.

model_joint.py: Full model architecture of GUIDE.

net_train_joint.py: The script to train the model using backpropagation.

Compare Random Graphs

The codes within this folder are used to train HG-ACN, and HG-DCN with random graphs.

wrapper.py: This python script is used to train the genetic branch using random graph embedding. We save the performance for each epoch which are later used.

find_checkpoint.m: Here we track the validation performance to find the early stopping point.

run_checkpoint_on_img_gen_data.py: Using the checkpoint model we evaluate the performance over the 208 test data as described in Section. 3.6 of manuscript.

run_checkpoint_on_training_img_gen_data.py: Using the checkpoint model we evaluate the performance over the training data.

net_train.py, net_test.py: These scripts used to train and test our model.

model_random.py: The model architecture of the genetic branch.

Bayes_vs_kshap

Guide_importance -> guide_topk_importance.py: This python script is used to identify top-K imaging features using Bayes importance. They are then used to mask the test data of each fold. The script saves the testing performance which is later compared with K-SHAP.

Shap_importance -> shap_importance.py: This python script is used to identify top-K imaging features using K-SHAP. This script stores the testing performance and the K-SHAP values of each feature which are later used for comparison, as described in Section. 3.6 of manuscript.

evaluate_performance.m: This script identifies the early stopping epoch for each fold based on lowest validation loss.

importance_compare.m: It compares and plots the testing performance as shown in Figure. 2 of manuscript.

guide_bayes_cross_cosine: Plots the pairwise cosine similarity across fold for Bayes feature selection scheme.

shap_cross_cosine: Plots the pairwise cosine similarity across fold for K-SHAP feature selection scheme.

Identifying Pathways

This set of codes are used to identify paths that exist between each root node and leaf node and are associated with the disorder.

wrapper.py: This python script train GUIDE over multiple random subsets of the data as explained in Section. 3.6 of the manuscript.

find_checkpoint.m: Here we track the validation performance to find the early stopping point.

extract_iterations.py: This python script uses the checkpoint models of GUIDE to store interaction scores between each pair node.

find_all_path.py: It finds and stores all possible paths that exist between each root node and each leaf node.

remaining_paths_to_run.py: It prepares the paths that will then fed to logistic regression model to identify potential discriminative pathways.

lrt_logistic_regression_multiple_pretrain.py: This takes in the patient specific interaction scores as input and runs logistic regression followed by likelihood ratio test. The p-values of the statistical test are stores for further evaluation.

pathway_importance.m: This script identifies statistically significant pathways.

embed_pathway_and_display.py: Embeds the pathway information using t-SNE and display 10 different categories of pathways.

Auxilliary Codes

convert_to_cpu.py: Sends a tensor to CPU.

convert_to_gpu.py: Sends a tensor to GPU.

convert_to_gpu_and_tensor.py: Converts an array to tensor and sends to GPU.

convert_to_gpu_scalar.py: Converts a scalar to tensor and sends to GPU.