Module Interface Specification for Re-ProtGNN

Yuanqi Xue

April 18, 2025

1 Revision History

Date	Version	Notes
Mar 19, 2025	1.0	Initial Draft

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/Yuanqi-X/Re-ProtGNN/blob/main/docs/SRS/SRS.pdf.

Contents

3 Introduction

The following document details the Module Interface Specifications for Re-ProtGNN, a re-implementation of an interpretable Graph Neural Network (GNN) Framework.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/Yuanqi-X/Re-ProtGNN/tree/main.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Re-ProtGNN.

Data Type	Notation	Description
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)
boolean	`bool`	Boolean value: either `True` or `False`
string	`str`	A sequence of Unicode characters
tensor	`Tensor`	A multi-dimensional array object from PyTorch
graph	`Data`	A graph object from PyTorch Geometric, with node and edge attributes
dataset	`Dataset`	A collection of graph objects for training or evaluation
dataloader	`DataLoader`	A PyTorch Geometric data loader for batching graph data
dictionary	`dict[K, V]`	A mapping from keys of type `K` to values of type `V`
list	`list[T]`	A sequence of elements of type `T`
function	`Customized Function`	A self-defined callable function

Re-ProtGNN uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding	
Behaviour-Hiding Module	Configuration Module Input Format Module Control Module Training Module Output Visualization Module
Software Decision Module	Model Module Inference Module Explanation Module Pytorch Module Pytorch Geometric Module GUI Module

Table 1: Module Hierarchy

6 MIS of Configuration Module

6.1 Module

Configuration

6.2 Uses

None

6.3 Syntax

6.3.1 Exported Constants

None

6.3.2 Exported Access Programs

None

6.4 Semantics

6.4.1 State Variables

- `data_args: DataParser` — Stores dataset-level configuration such as name, directory, splitting strategy, and seed.
- `model_args: ModelParser` — Stores GNN architecture settings and prototype-related parameters.
- `train_args: TrainParser` — Stores training hyperparameters including learning rate, batch size, and epoch count.
- `mcts_args: MCTSParser` — Stores Monte Carlo Tree Search and explanation-specific rollout parameters.
- `random_seed: int` — Stores the global seed used for generating random numbers.

6.4.2 Environment Variables

None

6.4.3 Access Routine Semantics

None - The state variables in this module are initialized when the system loads and are accessed directly by other modules using:

```
from utils.Configures import data_args, train_args, model_args, mcts_args
```

As such, no explicit accessor routines are exported.

6.4.4 Local Functions

DataParser(name: str, dir: str, split: list[$\mathbb{R}$], seed: int) $\rightarrow$ DataParser

- output: Returns a configuration object for dataset settings including `name`, `dir`, `split`, and `seed`.

ModelParser(model_name: str, hidden_dim: $\mathbb{N}$, num_prototypes: $\mathbb{N}$) $\rightarrow$ ModelParser

- output: Returns a configuration object containing the GNN model name, hidden dimension, and prototype count.

TrainParser(batch_size: $\mathbb{N}$, lr: $\mathbb{R}$, epochs: $\mathbb{N}$) $\rightarrow$ TrainParser

- output: Returns a configuration object with the training hyperparameters: `batch_size`, `lr`, and `epochs`.

MCTSParser(num_rollouts: $\mathbb{N}$, exploration_const: $\mathbb{R}$) $\rightarrow$ MCTSParser

- output: Returns a configuration object specifying the number of rollouts and exploration constant for MCTS-based explanation.

7 MIS of Input Format Module

7.1 Module

dataUtils

7.2 Uses

PyTorch Geometric Module (15), PyTorch Module (14), Configuration Module (6), Output Visualization Module (10)

7.3 Syntax

7.3.1 Exported Constants

None

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
load_dataset	-	`tuple[Dataset, int, int, dict[str → DataLoader]]`	`FileNotFoundError`, `ValueError`, `NotImplementedError`

7.4 Semantics

7.4.1 State Variables

None

7.4.2 Environment Variables

- `dataset_dir: str` — Filesystem path to the dataset root directory, obtained from `data_args.dataset_dir` defined in the Configuration Module.
- `log_file: str` — Path to the log file used by the `append_record()` routine exported from the Output Visualization Module.

7.4.3 Access Routine Semantics

load_dataset():

- transition:
  - Loads the dataset using `data_args.dataset_name` and `data_args.dataset_dir`, where `data_args` are defined in the Configuration Module.

- Logs the dataset name using `append_record(data_args.dataset_name)`, where `append_record()` is a routine exported from the Output Visualization Module

- output:
  - Returns a tuple: (`dataset`, `input_dim`, `output_dim`, `dataloader`) where:
    - `dataset`: graph dataset object loaded using `_get_dataset()`
    - `input_dim`: number of node features from `dataset.num_node_features`
    - `output_dim`: number of output classes from `dataset.num_classes`
    - `dataloader`: dictionary of DataLoaders split via `_get_dataloader()`

- exception:
  - `FileNotFoundError`: Raised if required dataset files are missing in the specified directory, such as missing raw '.pkl' or '.txt' files for the dataset.
  - `ValueError`: Raised if raw data files exist but are empty or malformed (e.g., missing node labels).
  - `NotImplementedError`: Raised if `data_args.dataset_name` does not match any supported dataset (i.e., not MUTAG, BA_2Motifs, or a MoleculeNet dataset).

7.4.4 Local Functions

_get_dataset(dataset_dir: str, dataset_name: str) → Dataset

- output: Selects an appropriate dataset loader based on `dataset_name` and returns the resulting dataset loaded from `dataset_dir`. See the Pytroch Geometric Module 15 for the type `Dataset`.

_get_dataloader(dataset: Dataset, batch_size: $\mathbb{N}$, data_split_ratio: list[$\mathbb{R}$]) → dict[str → DataLoader]

- output: Splits the input `dataset` into train/eval/test sets according to `data_split_ratio`, and returns DataLoaders batched by `batch_size`. See the PyTorch Geometric Module 15 for the type `DataLoader`.

8 MIS of Control Module

8.1 Module

Main

8.2 Uses

Hardware-Hiding Module, Configuration Module (6), Input Format Module (7), Model Module (11), Training Module (9), Inference Module (12), Explanation Module (13), Output Visualization Module (10)

8.3 Syntax

8.3.1 Exported Constants

None

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
main	clst: $\mathbb{R}$, sep: $\mathbb{R}$	-	`RuntimeError`

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Environment Variables

- Filesystem: the file system for loading/saving checkpoints and writing logs/images.
- GPU/CPU hardware for model training and inference.

8.4.3 Assumptions

None

8.4.4 Access Routine Semantics

main(clst, sep):

- transition: Loads the dataset and splits it into training, validation, and test sets. Initializes the GNN model and passes it to the `train` function for optimization. After training, the `test` function evaluates the model on the test set. Finally, explanation plots are generated and saved.
- output: None
- exception:
  - `RuntimeError`: if device mismatch or model loading fails.

8.4.5 Local Functions

None

9 MIS of Training Module

9.1 Module

Train

9.2 Uses

Hardware-Hiding Module, Configuration Module (6), Model Module (11), Explanation Module (10), Output Visualization Module (10)

9.3 Syntax

9.3.1 Exported Constants

None

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
train	model: `GnnNets`, dataset: `Dataset`, dataloader: `dict[str` $\rightarrow$ `DataLoader]`, clst: $\mathbb{R}$, sep: $\mathbb{R}$	-	`FileNotFoundError`

9.4 Semantics

9.4.1 State Variables

None

9.4.2 Environment Variables

Filesystem: the file system for saving model checkpoints.

9.4.3 Assumptions

None

9.4.4 Access Routine Semantics

train(model, dataset, dataloader, clst, sep):

- transition: Trains the model using the provided data and hyperparameters. Projects prototypes periodically. Monitors evaluation accuracy, saves the best-performing model to disk.
- output: None
- exception:
  - `FileNotFoundError`: if the dataset path or checkpoint directory is invalid

9.4.5 Local Functions

evaluate(loader: `DataLoader`**, model:** `GnnNets`**, criterion:** `Customized Function`**)** → `dict[str → float]`

- transition: None
- output: A dictionary containing the average loss and accuracy over the input dataset split. Specifically:
  - `"loss"`: average loss (float)
  - `"acc"`: classification accuracy (float)
- exception: None

10 MIS of Output Visualization Module

10.1 Module

OutputVisualize

10.2 Uses

Hardware-Hiding Module

10.3 Syntax

10.3.1 Exported Constants

None

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
PlotUtils	dataset_name: `str`	`PlotUtils instance`	-
plot	graph: `Data`, nodelist: `list[int]`, figname: `str`, kwargs: `dict`	-	-
append_record	info: `str`	-	`FileNotFoundError`

10.4 Semantics

10.4.1 State Variables

None

10.4.2 Environment Variables

- Filesystem: the file system for saving log files and outputting explanation images.

10.4.3 Assumptions

None

10.4.4 Access Routine Semantics

PlotUtils(dataset_name):

- transition: None
- output: A `PlotUtils` object with methods for graph visualization.
- exception: None

plot(graph, nodelist, figname, kwargs):

- transition: Generates explanation images and saves them to the specified path.
- output: None
- exception: None

append_record(info):

- transition: Appends the `info` string to the log file located in the given log directory.
- output: None
- exception:
  - `FileNotFoundError`: if the directory does not exist.

10.4.5 Local Functions

None

11 MIS of Model Module

11.1 Module

GnnNets

11.2 Uses

None

11.3 Syntax

11.3.1 Exported Constants

None

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
GnnNets	input_dim: `int`, output_dim: `int`, model_args: `dict`	`GnnNets`	`NotImplementedError`
forward	data: `Data`, prot-gnn_plus: `bool`, similarity: `Tensor`	logits: `Tensor`, prob: `Tensor`, emb1: `Tensor`, emb2: `Tensor`, min_distances: `Tensor`	-
update_state_dict	state_dict: `dict`	-	-
to_device	-	-	-

11.4 Semantics

11.4.1 State Variables

- `self.model`: the internal GNN encoder consisting of learnable layers.
- `self.prototype_vectors`: a tensor containing learnable prototype embeddings, where each prototype represents a latent concept tied to a specific class.
- `self.device`: the computing device (e.g., 'cpu' or 'cuda') on which the model is running.

11.4.2 Environment Variables

GPU/CPU hardware for model training and inference.

11.4.3 Assumptions

None

11.4.4 Access Routine Semantics

GnnNets(input_dim, output_dim, model_args):

- transition: None
- output: Returns an instance of the `GnnNets` class with specified input/output dimensions and model hyperparameters.
- exception:
  - `NotImplementedError`: if the specified model name in `model_args` is unsupported.

forward(data, protgnn_plus, similarity):

- transition: Moves graph data to the correct device and performs a forward pass through the model.
- output:
  - `logits`: raw output scores for each class.
  - `prob`: predicted class probabilities for each input graph, obtained by applying softmax to logits.
  - `emb1`: intermediate representation from an early layer of the model.
  - `emb2`: deeper-level embedding capturing higher-level graph features after additional processing layers.

  - `min_distances`: for each input graph, the minimum distance to each prototype vector.

- exception: None

update_state_dict(state_dict):

- transition: Loads and updates model parameters from a dictionary of saved weights.
- output: None
- exception: None

to_device():

- transition: Moves all model components to the device.
- output: None
- exception: None

11.4.5 Local Functions

None

12 MIS of Inference Module

12.1 Module

Test

12.2 Uses

Model Module (11), Output Visualization Module (10)

12.3 Syntax

12.3.1 Exported Constants

None

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
test	model: `GnnNets`, dataloader: `DataLoader`	-	`RuntimeError`

12.4 Semantics

12.4.1 State Variables

None

12.4.2 Environment Variables

None

12.4.3 Assumptions

The model has been trained and its best checkpoint has been loaded.

12.4.4 Access Routine Semantics

test(model, dataloader):

- transition: Evaluates the trained model on the test set. Computes loss and accuracy, and uses the Output Visualization Module to log results.
- output: None
- exception:
  - `RuntimeError`: if inference fails due to an invalid model state or shape mismatch

12.4.5 Local Functions

None

13 MIS of Explanation Module

13.1 Module

Explanation

13.2 Uses

Configuration Module (6)

13.3 Syntax

13.3.1 Exported Constants

None

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
get_explanation	data: `Data`, gnnNet: `GnnNets`, prototype: `Tensor`	coalition: `list[int]`, P: $\mathbb{R}$, embedding: `Tensor`	-

13.4 Semantics

13.4.1 State Variables

None

13.4.2 Environment Variables

None

13.4.3 Assumptions

None

13.4.4 Access Routine Semantics

get_explanation(data, gnnNet, prototype):

- transition: None
- output:
  - `coalition`: list of node indices forming the explanation.
  - `P`: float score indicating similarity to the prototype.
  - `embedding`: matrix of floats representing the masked subgraph embedding.
- exception: None

13.4.5 Local Functions

MCTSNode(coalition: `list[int]`**, data:** `Data`**, ori_graph:** `networkx.Graph`**, c_puct:** $\mathbb{R}$**, W:** $\mathbb{R}$**, N:** $\mathbb{R}$**, P:** $\mathbb{R}$**)** $\rightarrow$ `MCTSNode`

- transition: None
- output: A node object representing a state in the search tree.
- exception: None

mcts_rollout(tree_node: `MCTSNode`, **state_map:** `dict`, **data:** `Data`, **graph:** `networkx.Graph`, **score_func:** `Customized Function`) $\rightarrow \mathbb{R}$

- transition: None
- output: Scalar value representing the reward from this rollout.
- exception: None

child_scores(score_func: `Customized Function`, **children:** `list[MCTSNode]`) $\rightarrow$ `list[`$\mathbb{R}$`]`

- transition: None
- output: List of float scores, one for each child.
- exception: None

prot_score(coalition: `list[int]`, **data:** `Data`, **gnnNet:** `GnnNets`, **prototype:** `Tensor`) $\rightarrow \mathbb{R}$

- transition: None
- output: A float similarity score (higher = more aligned with prototype).
- exception: None

14 MIS of PyTorch Module

14.1 Module

Torch

14.2 Uses

None

14.3 Syntax

14.3.1 Exported Constants

None

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
Tensor	shape: `list[int]`, dtype: `str`	`Tensor`	-
cross_entropy	logits: `Tensor`, labels: `Tensor`	`Tensor`	-
Adam	parameters: `iterable`, lr: $\mathbb{R}$	`Optimizer`	-

14.4 Semantics

14.4.1 State Variables

None

14.4.2 Environment Variables

None

14.4.3 Assumptions

None

14.4.4 Access Routine Semantics

Tensor(shape, dtype):

- output: Returns a tensor initialized with zeros of the given `shape` and `dtype`.

cross_entropy(logits, labels):

- output: Computes the cross-entropy loss between `logits` and `labels`.

Adam(parameters, lr):

- output: Returns an Adam optimizer configured with the given `parameters` and learning rate `lr`.

14.4.5 Local Functions

None

15 MIS of PyTorch Geometric Module

15.1 Module

PyG

15.2 Uses

Torch

15.3 Syntax

15.3.1 Exported Constants

None

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
Data	x: `Tensor`, edge_index: `Tensor`	`Data`	-
MoleculeNet	root: `str`, name: `str`	`Dataset`	`FileNotFoundError`
DataLoader	dataset: `Dataset`, batch_size: $\mathbb{N}$	`DataLoader`	-
to_networkx	data: `Data`	`networkx.Graph`	-

15.4 Semantics

15.4.1 State Variables

None

15.4.2 Environment Variables

None

15.4.3 Assumptions

None

15.4.4 Access Routine Semantics

Data(x, edge_index):

- output: Constructs and returns a PyG graph object using `x` as node features and `edge_index` as edge indices.

MoleculeNet(root, name):

- output: Loads the dataset specified by `name` from directory `root` and returns a `Dataset` object.
- exception: `FileNotFoundError` if `root` does not exist.

DataLoader(dataset, batch_size):

- output: Returns a `DataLoader` that batches data from the given `dataset` with batch size `batch_size`.

to_networkx(data):

- output: Converts the input PyG `data` object into a NetworkX graph.

15.4.5 Local Functions

None

16 MIS of GUI Module

16.1 Module

Matplotlib

16.2 Uses

None

16.3 Syntax

16.3.1 Exported Constants

None

16.3.2 Exported Access Programs

Name	In	Out	Exceptions
axis	axis_choice: `str`	-	-
title	title_sentence: `str`	-	-
save_fig	figname: `str`	-	`FileNotFoundError`
close	choice: `str`	-	-

16.4 Semantics

16.4.1 State Variables

None

16.4.2 Environment Variables

- `figure_path: str` — Path where the current figure will be saved.
- `axis_visible: bool` — Whether axes are displayed in the active figure.
- `figure_title: str` — Title of the current figure.
- `figure_open: bool` — Whether there are any open figures.

16.4.3 Assumptions

None.

16.4.4 Access Routine Semantics

axis(axis_choice):

- transition: If `axis_choice == 'off'`, sets `axis_visible := False` and disables axes using `plt.axis('off')`. Otherwise sets `axis_visible := True`.

title(title_sentence):

- transition: Sets `figure_title := title_sentence` and updates the title of the current figure using `plt.title()`.

save_fig(figname):

- transition: Sets `figure_path := figname` and saves the current figure to the specified path using `plt.savefig(figname)`.
- exception: `FileNotFoundError` if `figname` refers to a non-existent directory.

close(choice):

- transition: Closes all active figure windows using `plt.close(choice)` and sets `figure_open := False`.

16.4.5 Local Functions

None

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.