Reflection and Traceability Report on Re-ProtGNN

Yuanqi Xue

This document outlines the development process of Re-ProtGNN, including revisions made in response to feedback from Dr. Smith and Yinying, the challenges encountered during implementation, the solutions applied, and the insights gained from the experience.

1 Changes in Response to Feedback

This project received feedback from two reviewers: the course instructor, Dr. Smith, and a peer reviewer, Yinying.

I incorporated Dr. Smith's feedback by reviewing the annotated PDF and committing updates to the documentation accordingly. Each suggestion was carefully addressed through revisions reflected in the related commits.

For the peer review, I responded to every GitHub issue raised by Yinying. Each issue was either resolved or clarified, and I included explanations along with links to the corresponding commits before closing the issues.

Below is a summary of the feedback and the changes made. Some commits listed are intermediate steps and may have been refined in the final documentation.

1.1 SRS

The following feedback was received on the SRS. Each item has been addressed with a corresponding update, commit link, or an explanation where applicable.

Feedback from Dr. Smith:

1. Use `\citep` for parenthetical citations.
   Update: commit b11412a: Applied `\citep` for all references.

2. Use b instead of b for math terms.
   Update: commit 63f44b3: Wrapped all math variables like `b` in math mode.

3. Section 1.4 (Mathematical Notation): Use non-italic font for function names `num_rows` and `num_cols`.

Update: commit 8bfcd9b: Changed font style for function names to upright.
Update: commit 4f1a6ef: Fixed missed instance in a later table.

4. Section 1.4 (Mathematical Notation): ReLU definition is circular.
   Update: commit 8bfcd9b: Reworded ReLU definition to avoid circularity.

5. Section 2 (Introduction): Add examples of potential classifications; Have a grammar issue.
   Update: commit f347818: Added example classifications, and fixed the grammar issue.

6. Section 2.2 (Scope of Requirements): Clarify whether input graphs are validated.
   Update: commit a0029b7: Modified scope of requirements & added input graph validity test in VnV

7. Section 2.3 (Characteristics of Intended Reader): Remove implementation details such as specific libraries.
   Update: commit 2c70c32: Removed references to programming languages and libraries.

8. Section 2.3: Remove assumption of gradient-based optimization knowledge.
   Update: commit 2c70c32: Clarified that concepts like partial derivatives are not assumed for high school readers.

9. Section 3.3 (System Constraints): This is an assumption, not a constraint.
   Update: commit b5abdba: Removed assumption from constraints section.

10. Section 4.1.1 (Terminology and Definitions): Expand dataset name.
    Explanation: After literature review, MUTAG has no formal expansion but is widely understood to refer to mutagenic aromatic compounds.

11. Section 4.3.3 (Theoretical Models, TM1): Include dimensions in the type declaration. Clarify input and output types of ReLU.
    Update: commit b6fa0b2: Included tensor dimensions in TM1 type declarations. Clarified ReLU takes real-valued input and returns non-negative tensor.

12. Section 4.3.4 (General Definitions, GD2): Some symbols are undefined.
    Update: commit 8f7c3b6: Defined all missing symbols used in GD2.

13. Section 4.3.4 (General Definitions, GD3): Type is missing.
    Update: commit e2b43b9: Added type declarations with dimensions for variables in GD3.

14. Section 4.3.4 (General Definitions): Reorganize theory to introduce optimization goals before methods.
    Update: commit dac6a18: Added a new General Definitions (i.e., GDL) to

bridge gap in General Definitions and to state loss minimization objective before introducing gradient methods.

15. Section 5.2 (Nonfunctional Requirements - Accuracy): Clarify what it is compared against.
Update: commit a0ff591: Rephrased Accuracy to Reliability and updated Traceability Matrix.

16. Section 5.2 (Usability): Currently implementation plan, not a usability definition.
Update: commit a0ff591: Rewrote usability as measurable NFR (e.g., survey).

17. Section 5.2 (Maintainability): Currently an implementation detail, not a requirement.
Update: commit a0ff591: Removed maintainability from NFRs and updated Traceability Matrix.

18. Traceability Matrix: NFRs usually trace to most components.
Update: commit a0ff591: Expanded NFR trace links in the matrix.

Feedback from Yinying:

1. Section 1.2 (Table of Symbols): Group all entries related to the adjacency matrix together to improve clarity
Update: commit 3026198: Rearranged the table entries in Section 1.2 so that all symbols related to the adjacency matrix are grouped together for readability.

2. Section 3.1 (System Context): The diagram shows the user providing graphs as input, while the text says the user downloads and loads the MUTAG dataset. This creates inconsistency
Update: commit 1579544: Clarified the text to explain that MUTAG is a dataset of graphs, aligning the textual description with the context diagram.

3. Section 3.3 (System Constraints): The described item does not limit the system design and should be classified as an assumption instead
Update: commit b5abdba: Removed the assumption from system constraint, consistent with similar feedback from Dr. Smith.

4. Section 4.1.1 (Terminology and Definitions): The MUTAG dataset may not be a terminology or definition; consider relocating it
Explanation: I opted to retain the MUTAG dataset in this section as a definition, since it improves clarity for readers unfamiliar with the dataset and aligns with the structure of the document.

5. Section 4.3.2 (Assumptions): Consider removing justification phrases such as "as required by 4.3.3 Theoretical Models."

Explanation: I opted to keep this reference, as linking assumptions directly to the corresponding theoretical models improves traceability and clarity for the reader.

1.2 Design and Design Documentation

The following feedback was received on the Module Guide (MG). Each item has been addressed with a corresponding update, commit link, or an explanation where applicable.

Feedback from Dr. Smith (MG):

1. There is normally a module to provide GUI services
   Update: commit 110e5af: Added a GUI module to the MG to handle output visualization responsibilities. Also incorporated PyTorch and PyTorch Geometric modules as suggested in related MIS feedback.

2. Use hierarchy figure: Use curved lines to avoid crossing the boxes
   Update: commit c63b469: Redrew the Use Hierarchy diagram using curved arrows to improve readability and ensure lines do not cross any module boxes.

3. Use hierarchy figure: Why does Inference depend on Output Visualization?
   Update: commit 69eed93: The Inference module depends on Output Visualization because loss and accuracy values are logged during inference to monitor performance in log files. Updated the service description of the Training Module in Section 7.2.4 to reflect this more clearly.

Feedback from Yinying (MG):

1. Section 7.2.4: Clarify why the Training Module depends on the Output Visualization Module
   Update: commit 69eed93: In response to Feedback 3 from Dr. Smith, the service description of the Training Module in Section 7.2.4 was updated to clarify that it depends on the Output Visualization Module to log loss and accuracy during training.

2. Section 7.2.2: Incorrect module reference and vague service description
   Update: commit b72861c: Replaced the outdated phrase "input parameters module" with a more accurate service description. The updated line now specifies that the module "loads and parses datasets into standardized graph representations using PyTorch Geometric data structures."

The following feedback was received on the Module Guide (MIS). Each item has been addressed with a corresponding update, commit link, or an explanation where applicable.

Feedback from Dr. Smith (MIS):

1. Section 4 (Notations): This table is for mathematical notation. If PyTorch types are used, they should be abstracted or moved to a module
   Update: commit b2f2500: Revised Section 4 to remove PyTorch-specific types and express all type declarations using mathematical notation.

2. Section 5 (Module Decomposition): Which module represents PyTorch?
   Update: commit b06753a: Added PyTorch and PyTorch Geometric modules to the decomposition and included the module specifications. Also included a GUI module per MG feedback.

3. Start each module on a new page
   Update: commit 5e23fef: Inserted page breaks so that each module starts on a new page.

4. Sections 6.3.1, 6.4.2, and 6.4.4: `DataParser` is undefined; constants seem to be variables; environment variable types are missing and unused; container access is unclear
   Update: commit a304222: Moved arguments to state variables, defined `DataParser` as a local function, and clarified container access through imports.

5. Sections 7.3.2 and 7.4.4: Where is `Dataset` defined? Access routine specifications are missing method arguments; `AssertionError` in `get_dataloader` is too vague
   Update: commit 5b145b8: Updated function specs to include method arguments and clarified exceptions. Replaced `get_dataset` and `get_dataloader` with `load_dataset`, and defined output types in PyTorch Geometric module.

6. Section 7.3.2: Clarify interpretation of `dict[str -> DataLoader]`
   Update: commit ec047c2: Added a definition for the type notation `dict[K → V]` to Section 4.

7. Section 8 (Control Module): What state or environment variables are being modified?
   Update: commit dd44c91: Specified transitions on relevant environment variables within the Control Module.

8. Section 9 (Training Module): Method arguments are missing in access routine semantics
   Update: commit a535ac4: Rewrote access routine specifications to explicitly include all method arguments.

9. Section 10 (Output Visualization Module): `Filesystem` environment variable is not used in any transitions
   Update: commit c9f06a8: Updated environment variables and transitions so that every environment variable was explicitly used.

10. Section 11 (Model Module): Use ragged-right alignment to improve table readability
    Update: commit 1579525: Applied ragged-right alignment to the exported access program table for improved formatting.

Feedback from Yinying (MIS):

1. Section 4 (Notation): The type `string` is defined as a sequence of characters, but `character` is not defined
   Update: commit 84dcbd4: Added `character` as a defined type in the Notation table to support the definition of `string`.

2. Section 4 (Notation): `dataloader` seems more like a function or class instance, not a type
   Update: commit b06753a: Clarified the use of `DataLoader`, `Dataset`, and `Data` by adding a PyTorch Geometric Module that defines these as external object types originating from the PyG library.

3. Section 10.3.2 and 10.4.4: Distinction between `PlotUtils instance` and `PlotUtils object` is unclear
   Update: commit 8debb26: Updated all references to use consistent terminology, referring to class-based entities as "objects" rather than mixing with "instances," and clarified the output types accordingly.

1.3 VnV Plan

The following feedback was received on the Verification and Validation (VnV) Plan. Each issue will be addressed with an appropriate update and commit link.

Feedback from Dr. Smith:

1. Title: Use the correct program name (ProgName)
   Update: commit 4e72d42: Updated the document title to correctly reflect ProgName.

2. Section 2.2 (Objectives): It is easier to describe the accuracy you achieve rather than a prior specified target
   Update: commit a7fe22e: Removed Accuracy from Objectives to avoid predefining it as a target metric.

3. Use `\citep{}` for parenthetical citations
   Update: commit 60ca569: Replaced `\cite` with `\citep` throughout the VnV Plan.

4. Section 2.3 (Challenge Level and Extras): What extra would be included?
   Update: commit ab82a69: Clarified the absence of extras and rephrased the section accordingly.

5. Section 2.4 (Relevant Documentation): Add MG, MIS, and VnV Plan
   Update: commit 10f89fb: Added MG, MIS, and VnV Plan to the relevant documentation list.

6. Section 3.2–3.4: Add link or citation to the SRS, MG, MIS, and VnV checklist
   Update: commit e4b0545: Added checklist references for SRS, MG, MIS, and VnV to Sections 3.2–3.4.

7. Section 3.5 (Implementation Verification Plan): Presentation time isn't enough for a typical code walkthrough
   Update: commit 3b05733: Removed the code walkthrough due to time constraints in presentation.

8. Section 3.6 (Automated Testing and Verification Tools): "Analyzed how?"
   Update: commit 35cea6a: Removed regression test analysis after considering randomness in model behavior.

9. Section 3.7 & 4.1: Add link or citation to the dataset
   Update: commit 2c32201: Added dataset citation in Software Validation and Functional Testing sections.

10. Section 4.1.1 (Input Verification): Unclear test specification – someone else should be able to execute your plan
    Update: commit 62a53ef: Rewrote input verification with three detailed test cases (missing file, format consistency, and valid dataset load).

11. Section 4.1.2 (Model Training Verification): Ambiguity – test passes based on what?
    Update: commit 62a53ef: Revised training verification to include pass/fail criteria based on loss convergence and training stability.

12. Section 4.1.3 (Inference Verification): Clarify dataset, formula, and expected output
    Update: commit 62a53ef: Added accuracy formula and identified test dataset and reference accuracy.

13. Section 4.2.1 (Accuracy): Specify dataset and how classification accuracy is calculated
    Update: commit 7879f77: To align with the SRS, Accuracy was removed as a nonfunctional requirement and replaced with Reliability. While this specific feedback no longer applies directly, it helped guide the revision of the Reliability test case, which is now written with clear criteria.

14. Section 4.2.2 (Usability): Be specific—how is feedback collected?
    Update: commit 7879f77: Updated Usability test to specify use of a user survey for evaluation.

15. Section 4.2.3 (Area of Testing 3 – Portability): How is this measured?
    Update: commit 7879f77: Portability was removed from the nonfunctional requirements due to complex dependencies on specific CUDA versions, compatibility issues with PyTorch Geometric, and the limited timeline of the project.

Feedback from Yinying:

1. Section 4.1.2 (Test for Training Loss Optimization): Use training dataset rather than full MUTAG, and graph-based output makes trend verification difficult
   Update: commit 62a53ef: Updated "T2: Loss Convergence Test" to use the training split of the dataset instead of the entire MUTAG set. Return a list of loss values and verify decreasing trend programmatically, replacing visual inspection of the curve.

2. Section 4.2.1 (T4 Accuracy Test): Traceability matrix and test label unclear
   Update: commit 7879f77: Replaced T4 Accuracy test with a Reliability test and updated the traceability matrix to reflect the proper links to R2 and R3.

3. Unit Test `test-M3-2`: Output is ambiguous — be specific about dictionary keys
   Update: commit 3fa35e8: Updated the test to explicitly specify the expected keys of the returned dictionary.

4. Unit Test `test-M6-3`: Input says "Same as above" — should restate all input values
   Update: commit 4e3e294: Revised `test-M6-3` to fully state all input parameters instead of referring back to previous tests.

2 Challenge Level and Extras

2.1 Challenge Level

Research Project

2.2 Extras

No applicable.

3 Design Iteration (LO11 (PrototypeIterate))

The design of Re-ProtGNN evolved significantly throughout the course of the project in response to formal feedback, implementation challenges, and iterative improvements in usability, modularity, and correctness.

The earliest version of the system lacked modular separation for core libraries and hardware-hiding layers. After feedback from Dr. Smith and subsequent MIS revisions, we introduced three new modules: PyTorch, PyTorch Geometric, and GUI, to isolate external library usage and visualization responsibilities. These changes helped clarify module dependencies and improved maintainability.

Another major change came from feedback on data handling. Originally, the Input Format Module did not clearly expose its internal logic or support reusable access routines. After Yinying's feedback and reflection on testability, I redesigned it to expose a single `load_dataset()` routine that handles preprocessing, validation, and returns data loaders. This change also enabled clearer specification and testing of its behavior in the VnV Plan.

The VnV Plan itself underwent substantial iteration. Initially, tests such as "Loss Convergence" relied on visual plots. Based on usability feedback, we rewrote the tests to return numeric values and assess convergence programmatically. We also reorganized test specifications to be self-contained, addressing peer feedback that descriptions like "same as above" were ambiguous to readers.

Finally, the use of types and notation was cleaned up iteratively. For example, the `dataloader` type was initially used without clear context. After suggestions from both reviewers, we updated the Notation table to define all mathematical notation and have modules for external libraries (e.g., PyTorch Geometric) to define library-specifc types. I also include a note to reference to section where types are defined, if feeling it would be ambiguous. The same refinement was applied to terminology like "PlotUtils instance," now uniformly referred to as "object" for consistency.

These changes were not only driven by reviewer suggestions but also by the evolving understanding of what would make the system easier to extend, verify, and explain.

4 Design Decisions (LO12)

Limitations. A key limitation was imposed by the project's reliance on PyTorch Geometric, which has known compatibility issues across CUDA versions and platform-specific behavior (e.g., CPU vs. GPU). This directly influenced the decision to remove "Portability" as a non-functional requirement in the SRS and VnV Plan, since meeting portability guarantees would require time and resource investments beyond scope.

Assumptions. One core assumption was that all datasets (e.g., MUTAG) were available in standard format and accessible via PyTorch Geometric. This assumption was encoded in the theoretical models and tested in VnV (e.g., T3: Valid Dataset Load Test) before inputting the data into the model. We also assumed the training loss would decrease for a well-behaved model, which informed the design of the Loss Convergence Test.

Constraints. Due to project deadlines, we prioritized modular readability and documentation correctness over performance optimization. This is reflected in decisions such as implementing full parser functions for each configuration

class (e.g., `DataParser`, `TrainParser`), rather than relying on external YAML parsing. Additionally, we constrained our model backbone to GCN for stability across runs, even though more expressive architectures like GAT or GIN were briefly considered.

Design trade-offs. In designing the Output Visualization Module, we chose to log accuracy and loss to file rather than render interactive plots. This decision improved compatibility with automated testing tools, even though it reduced usability during real-time training.

The design decisions were refined throughout iterative feedback and guided by both usability and testability principles.

5 Reflection on Project Management (LO24)

5.1 What Went Well?

One of the most satisfying aspects of this project was successfully integrating GitHub Actions for automated testing and verification. Although GitHub-hosted runners only support CPU-based execution and lack compatibility with GPU-dependent components (like specific CUDA versions required by PyTorch Geometric), I was able to overcome this limitation by configuring a self-hosted GitHub Actions runner on my local development server. This setup allowed me to run all system and unit tests within a consistent environment, using the exact versions of CUDA and PyTorch needed for reproducibility.

5.2 What Went Wrong?

The system's final performance was lower than initially anticipated, particularly in terms of accuracy consistency across multiple runs. Due to the stochastic nature of training deep models on small datasets like MUTAG, the accuracy metric varied significantly between runs, even when using fixed seeds. As a result, I had to remove "Accuracy" as a non-functional requirement in the SRS and VnV Plan and replace it with a more robust measure of "Reliability," which better reflects the correctness of the system. This was a necessary adjustment but also a reminder of the importance of setting realistic and measurable goals for evaluation.

5.3 What Would You Do Differently Next Time?

For future projects, especially those involving machine learning pipelines, I would define evaluation criteria more conservatively upfront and anticipate variance in model performance during the requirements specification phase. I would also allocate time earlier in the project timeline to experiment with CI pipeline configurations, particularly when GPU access or hardware-specific dependencies are involved. This would allow me to ensure CI/CD is working from the beginning instead of troubleshooting later. Lastly, I would formalize more rigorous baseline testing earlier on to determine if higher-performing architectures

(e.g., GAT, GIN) are worth the additional complexity, or if simpler, more stable backbones like GCN should remain the default.

One area I struggled with during this project was time management. I often found myself submitting components close to the deadline or needing to seek for extensions. This was partly due to the overall workload of the semester, but also because I underestimated the time required for debugging and refining documentation. For future projects, I plan to set internal deadlines at least two days before the official ones to leave room for unexpected issues or last-minute revisions. I also intend to better track progress across subtasks to avoid bottlenecks near submission.