

System Verification and Validation Plan for Live Neuro

Bo Liang

February 26, 2025

Revision History

Date	Version	Notes
February26, 2025	1.0	Initial Draft

Contents

Table of Contents

1. SYMBOLS, ABBREVIATIONS, AND ACRONYMS	3
2. GENERAL INFORMATION	5
2.1 SUMMARY	5
2.2 OBJECTIVES	5
2.3 CHALLENGE LEVEL AND EXTRAS	6
2.4 RELEVANT DOCUMENTATION	7
3. PLAN.....	7
3.1 VERIFICATION AND VALIDATION TEAM	7
3.2 SRS VERIFICATION PLAN	7
3.3 DESIGN VERIFICATION PLAN.....	8
3.4 IMPLEMENTATION VERIFICATION PLAN	8
3.5 AUTOMATED TESTING AND VERIFICATION TOOLS	8
3.6 SOFTWARE VALIDATION PLAN.....	9
4. SYSTEM TESTS.....	9
4.1 TESTS FOR FUNCTIONAL REQUIREMENTS.....	9
4.2 TESTS FOR NONFUNCTIONAL REQUIREMENTS.....	10
4.3 TRACEABILITY BETWEEN TEST CASES AND REQUIREMENTS	13
5. UNIT TEST DESCRIPTION	14
5.1 UNIT TESTING SCOPE	14
5.2 TESTS FOR FUNCTIONAL REQUIREMENTS.....	14
5.3 TESTS FOR NONFUNCTIONAL REQUIREMENTS.....	14
5.4 TRACEABILITY BETWEEN TEST CASES AND MODULES	16
6. USABILITY SURVEY QUESTIONNAIRE.....	17

1. Symbols, Abbreviations, and Acronyms

The definition of symbols, abbreviations and acronyms is as same as those in my SRS documents.

This document presents the Verification and Validation (VnV) Plan for Live Neuro, an interactive neural data visualization tool. The plan is structured into four main sections:

1. General Information

- Provides an overview of Live Neuro, its objectives, challenge level, and relevant documentation.
- Defines the scope of verification, including functional and non-functional requirements.

2. Plan

- Describes the verification strategies for different project phases: SRS, Design, VnV, and Implementation.
- Details the roles and responsibilities of the Verification and Validation Team.
- Includes plans for automated testing, validation tools, and software validation.

3. System Tests

- Verifies both functional and non-functional requirements at the system level.
- Defines test cases for acoustic stimulus processing, interactive visualization, and cross-platform compatibility.
- Ensures the system behaves as expected through automated and manual testing.

4. Unit Test Description

- Focuses on validating individual modules such as data preprocessing, TRF model computation, and chart rendering.
- Covers both functional and non-functional aspects with unit test cases.
- Includes traceability analysis to ensure proper coverage of requirements.

This VnV plan provides a structured and comprehensive approach to verifying and validating Live Neuro, ensuring reliability, accuracy, and usability before deployment.

2. General Information

2.1 Summary

Live Neuro is an interactive neural data visualization tool designed to display EEG and MEG data through linked multidimensional charts. It supports continuous response analysis for acoustic and semantic stimuli. This plan aims to verify its functional correctness, interactivity, cross-platform compatibility, and user-friendliness.

2.2 Objectives

Primary Objectives:

- Verify the implementation of functional requirements and non-functional requirements.
- Ensure that data visualization results meet scientific standards (NFR1).
- Validate cross-platform compatibility (Linux, Windows 10+, macOS).

Non-Objectives:

- Do not validate the internal implementation of third-party libraries (e.g., Matplotlib).
- Do not conduct large-scale performance stress testing due to resource limitations.

Exclusions and Rationale

Certain areas have been explicitly excluded from testing due to well-defined constraints. Below is a list of these exclusions and their justifications:

1. Third-Party Library Internal Implementation:

- Justification: The project relies on established third-party libraries (e.g., Matplotlib) for visualization, which have been extensively tested and validated by their respective development communities.
- Risk Mitigation: Instead of testing the internal workings of these libraries, the focus is on ensuring their correct integration within Live Neuro.

2. Large-Scale Stress Testing:

- Justification: Due to resource limitations, Live Neuro will not undergo high-volume performance stress testing (e.g., handling extremely large datasets beyond typical neuroscience use cases).

- Risk Mitigation: Instead, performance tests will focus on optimizing real-time processing within realistic workload scenarios to ensure smooth operation in practical research settings.
3. **Hardware-Specific Optimizations:**
- Justification: The project does not target specific hardware acceleration (e.g., GPU-based computations) beyond standard system configurations.
 - Risk Mitigation: The software will be tested for general computational efficiency across multiple platforms but without hardware-dependent optimizations.

2.3 Challenge Level and Extras

Challenge Level:

- Advanced (requires support for complex interactive charts and multi-platform adaptation).

Live Neuro is classified as an **Advanced** project due to several key factors that elevate its complexity:

1. **Technical Complexity:**
 - The project involves processing and visualizing high-dimensional neural data (EEG and MEG), which requires specialized signal processing techniques and efficient data handling.
 - Integration of multiple visualization libraries and frameworks to support interactive, multi-chart data representation.
2. **Performance Demands:**
 - The system must process large-scale time-series neural data in real time while ensuring smooth interactivity.
 - Optimization is necessary to maintain low latency in data rendering and avoid performance bottlenecks when handling high-resolution EEG/MEG signals.
3. **User Interface Challenges:**
 - The interactive visualizations require seamless linking between multiple charts to provide an intuitive exploration experience.
 - Implementation of user-friendly interfaces that allow researchers to dynamically manipulate and analyze data without extensive technical knowledge.
 - Cross-platform compatibility (Linux, Windows, macOS) adds additional UI/UX considerations to ensure consistency across different environments.

Additional Content:

- User usability testing (feedback from users with a neuroscience background).
- Static code analysis (using flake8 and pylint).

2.4 Relevant Documentation

- **SRS Document:** Defines requirements and system models.
- **User Manual:** Operation guide and test configuration instructions (to be supplemented).

3. Plan

3.1 Verification and Validation Team

Member	Role	Responsibilities
Bo Liang	Project developer	Develop software
Dr. Brodbeck	reviewer	Review structure of the project
Ziyang Fang	Code reviewer	Performs static code analysis and unit test coverage

3.2 SRS Verification Plan

Method:

- **Structured Review** (based on SRS Section 8 traceability matrix).

Steps:

1. Invite neuroscience researchers to review the main function of the tool.
2. Use a checklist to evaluate the main part of the document including Completeness, Feasibility, Traceability
3. Reviewers will provide feedback through github Issue

4. Author will address all the issues and conduct a final review of the document

3.3 Design Verification Plan

Method:

- Automated code test + Code Walkthrough + Design Document Review.

Tools:

- Doxygen for generating module dependency diagrams, combined with design documents to clarify dependencies between different modules.

3.4 Implementation Verification Plan

Unit Testing:

- pytest will be used as the main test tool and some other visualization libraries testing tool will be used too Such as `compare_image` function in `matplotlib` testing
- The Coverage goal $\geq 90\%$ (measured using `pytest-cov`).

Static Analysis:

- Use `flake8` to check code syntax and clean it based on the instructions provided
- Use `mypy` as static type checker to check type annotations to detect type errors without running the code.

3.5 Automated Testing and Verification Tools

Tool	Purpose
pytest	Functional and unit testing framework
flake8	check code style

mypy	type annotation verification
Matplotlib	Image baseline comparison (image_comparison)
GitHub Actions	Cross-platform CI/CD pipeline

3.6 Software Validation Plan

Method:User Acceptance Testing (UAT).

Steps:

1. Demonstrate core functionality to external supervisors (Rev 0 presentation).
2. Collect user feedback (usability survey in the appendix).

4. System Tests

4.1 Tests for Functional Requirements

Test Case1(High Priority):

FR:the interactive data plots generated by Live Neuro should be exactly identical to original plots

Input: calculation result of TRF model

Expected Output: data plots generated by Live Neuro should be exactly identical to original plots

Verification Method:image_comparison method in matplotlib testing

How test will be performed: this test will be automated by pytest

Test Case2(High Priority):

FR:the output of LiveNeuro should be interactive between different plots

Input: Selecting data points across linked charts.

Expected Output: Other charts highlight corresponding points simultaneously.

Verification Method: Use mplcursors to trigger events and compare data in the Corresponding plot to the input data

How test will be performed: this test will be executed by pytest and manually

4.2 Tests for Nonfunctional Requirements

4.2.1 Area of Testing1

Test Case3: Usability(Medium Priority)

- **Category:** Non-functional, User-Centered, Manual Evaluation
- **Initial Condition:** The Live Neuro application is fully operational and prepared for hands-on evaluation.
- **Test Inputs/Conditions:**
 - Participants engage with the software in a simulated neuroscience research workflow.
 - A structured usability survey (Appendix 6.2) is conducted, covering ease of interaction, visualization clarity, and feature intuitiveness.
- **Expected Outcome:**
 - Participants can efficiently navigate core functionalities with minimal guidance.
 - Survey responses highlight strengths and areas for enhancement.
- **Execution Steps:**
 1. Deploy Live Neuro to a group of neuroscience researchers and graduate students.
 2. Observe user behavior and gather feedback on usability pain points.
 3. Analyze survey data, identify usability trends, and refine the user interface accordingly.

Test Case4(Medium Priority): Maintainability

- **Category:** Non-Functional, Code Reliability, Automated Validation
- **Initial Condition:** The project contains distinct computational modules, each responsible for a separate functionality.
- **Test Inputs/Conditions:**
 - Run an automated test coverage analysis using pytest-cov.
- **Expected Outcome:**
 - The overall test coverage remains above a predefined threshold ($\geq 90\%$).
 - Any uncovered sections of code are identified and targeted for additional tests.
- **Execution Steps:**
 1. Execute all existing test cases with coverage tracking enabled.
 2. Generate a detailed report showing per-module and per-function coverage statistics.
 3. Identify weakly tested components and extend test coverage accordingly.

Test Case5(High Priority): Portability

- **Category:** Non-Functional, Deployment Readiness, Manual & Automated Testing
- **Initial Condition:** The Live Neuro source code is version-controlled and ready for multi-platform testing.
- **Test Inputs/Conditions:**
 - Attempt to install, build, and execute the software on Linux (Ubuntu 22.04), Windows 11, and macOS Ventura.
- **Expected Outcome:**
 - The software compiles without errors and runs smoothly on all tested platforms.

- Platform-specific discrepancies (if any) are documented and addressed.

- **Execution Steps:**

1. Configure test environments across different operating systems.
2. Build and execute Live Neuro, ensuring its core functionalities (e.g., data preprocessing, EEG/MEG visualization) behave consistently.
3. Log and analyze any platform-dependent issues, optimizing cross-platform performance as needed.

Test Case6(**Medium Priority**): Reusability

- **Category:** Non-Functional, Code Design Evaluation, Manual Review

- **Initial Condition:** Live Neuro's codebase consists of distinct functional components (e.g., data handling, signal processing, interactive visualization).

- **Test Inputs/Conditions:**

- Review module separation and cross-component dependencies.

- **Expected Outcome:**

- The software architecture follows clear separation of concerns, with reusable components structured for extensibility.

- **Execution Steps:**

1. Analyze the code structure, identifying modules that can be independently reused or extended.
2. Assess whether core computational functions are generalizable for broader neuroscience research applications.
3. Document reusability strengths and suggest refactoring where necessary to improve modularity.

4.3 Traceability Between Test Cases and Requirements

Test Case	Covered Requirements
Test Case1	R1
Test Case2	R2
Test Case3	NFR1
Test Case4	NFR2
Test Case5	NFR3
Test Case6	NFR4

4.4 Edge cases and failure Testing

To ensure robustness and reliability, Live Neuro will undergo rigorous edge case and failure scenario testing. This section outlines specific areas where unexpected behavior might occur and the strategies for addressing them.

1. Edge Case Testing:

- **Extreme Data Values:** Test how the system handles unusually large or small neural signal values, ensuring visualizations do not break or become unreadable.
- **Sparse Data Inputs:** Verify correct behavior when datasets contain missing values, ensuring the visualization does not crash or produce misleading results.
- **Unusual Data Formats:** Test how the system processes unexpected data formats, validating error handling and appropriate user feedback.

2. Failure Scenario Testing:

- **Unexpected User Inputs:** Ensure the application handles invalid user actions gracefully, such as selecting non-existent data points or attempting to visualize corrupted files.
- **Memory and Performance Failures:** Simulate scenarios where the system is under heavy computational load, ensuring it remains responsive or provides appropriate warnings.

- **System Crashes:** Intentionally trigger conditions that could lead to crashes (e.g., dividing by zero in computations) and ensure robust exception handling is in place.

5. Unit Test Description

5.1 Unit Testing Scope

Included Modules:

- data visualization.

Excluded Modules:

- Native Matplotlib functions (assumed verified).

5.2 Tests for Functional Requirements

UT-F1(High Priority): Interactive Visualization Sync

- **Objective:** Ensure multi-chart linking updates correctly when a user selects data points.
- **Test Type:** Functional, Automated
- **Test Input:** User selects a data point in one chart.
- **Expected Output:** The corresponding data point is highlighted in all linked charts.

5.3 Tests for Nonfunctional Requirements

UT-NF1(Low Priority): Usability – Interface Clarity and User Experience

- **Objective:** Ensure tooltips, labels, and messages are clear and helpful.
- **Test Type:** Non-Functional, Manual Inspection
- **Test Input:** Verify UI components like tooltips, axis labels, and error messages.

- **Expected Output:**

- All UI elements provide relevant and accurate feedback.
- User interactions result in expected system responses.

UT-NF2(Medium Priority): Maintainability – Code Readability and Structure

- **Objective:** Ensure the codebase follows maintainable coding practices.
- **Test Type:** Non-Functional, Static Analysis
- **Test Input:** Run Flake8 and Pylint on the project.
- **Expected Output:** No major code quality issues detected.

UT-NF3(Medium Priority): Maintainability – Test Coverage

- **Objective:** Ensure each module has sufficient test coverage.
- **Test Type:** Non-Functional, Automated
- **Test Input:** Run pytest with coverage tracking.
- **Expected Output:** Test coverage is at least **90%**.

UT-NF4(High Priority): Portability – Multi-Platform Compatibility

- **Objective:** Verify the software runs on Linux, Windows, and macOS.
- **Test Type:** Non-Functional, Automated
- **Test Input:** Detect OS type and verify execution.
- **Expected Output:** Live Neuro runs successfully on all platforms.

UT-NF5(Medium Priority): Reusability – Module Independence

- **Objective:** Ensure that major components are modular and reusable in other projects.
- **Test Type:** Non-Functional, Manual Review
- **Test Input:** Check for modularity in different code files.
- **Expected Output:**
 - Modules should not have unnecessary dependencies.
 - Core functions should be easily extractable and reusable.

5.4 Traceability Between Test Cases and Modules

Test Case	Covered Module
UT-F1	R1.R2
UT-NF1	NF1
UT-NF2	NF2
UT-NF3	NF2
UT-NF4	NF3
UT-NF5	NF4

6. Usability Survey Questionnaire

Section 1: General Experience

1. How easy was it to install and set up Live Neuro?

- ☐ 1 - Very difficult
- ☐ 2 - Somewhat difficult
- ☐ 3 - Neutral
- ☐ 4 - Somewhat easy
- ☐ 5 - Very easy

2. How intuitive was the user interface?

- ☐ 1 - Very confusing
- ☐ 2 - Somewhat confusing
- ☐ 3 - Neutral
- ☐ 4 - Somewhat intuitive
- ☐ 5 - Very intuitive

3. Were you able to complete basic tasks (e.g., loading data, visualizing neural activity) without external help?

- ☐ 1 - No, I could not complete the tasks
- ☐ 2 - No, I needed significant assistance
- ☐ 3 - Yes, but I struggled
- ☐ 4 - Yes, but with minor difficulties

- ☐ 5 - Yes, without any difficulty

Section 2: Visualization and Interaction

4. How clear and informative were the generated EEG/MEG visualizations?

- ☐ 1 - Very unclear
- ☐ 2 - Somewhat unclear
- ☐ 3 - Neutral
- ☐ 4 - Somewhat clear
- ☐ 5 - Very clear

5. Were the interactive features (e.g., zooming, selecting data points, multi-chart linking) responsive and easy to use?

- ☐ 1 - Very unresponsive and difficult to use
- ☐ 2 - Somewhat unresponsive
- ☐ 3 - Neutral
- ☐ 4 - Somewhat responsive and easy to use
- ☐ 5 - Very smooth and intuitive

6. Did the software provide sufficient feedback when interacting with the visualizations (e.g., tooltips, highlights, error messages)?

- ☐ 1 - No feedback at all
- ☐ 2 - Some feedback, but often unclear
- ☐ 3 - Neutral

- ☐ 4 - Mostly clear feedback
- ☐ 5 - Very clear and helpful feedback

Section 3: Performance and Reliability

7. How would you rate the speed of data processing and visualization?

- ☐ 1 - Extremely slow, unusable
- ☐ 2 - Slow but functional
- ☐ 3 - Neutral
- ☐ 4 - Acceptable speed
- ☐ 5 - Very fast

8. Did you experience any crashes, errors, or unexpected behaviors?

- ☐ 1 - Very frequent crashes/issues
- ☐ 2 - Frequent minor issues
- ☐ 3 - Neutral
- ☐ 4 - Rare minor issues
- ☐ 5 - No issues at all

Section 4: Cross-Platform Compatibility

9. How well did Live Neuro perform on your operating system?

- ☐ 1 - Did not work at all

- ☐ **2** - Many issues, barely usable
- ☐ **3** - Neutral
- ☐ **4** - Mostly functional with minor issues
- ☐ **5** - Fully functional without problems

10. Did you encounter any platform-specific issues?

- ☐ **1** - Major problems, software unusable
- ☐ **2** - Significant issues but still usable
- ☐ **3** - Neutral
- ☐ **4** - Minor platform differences
- ☐ **5** - No platform-related issues

Section 5: Documentation and Support

11. How helpful was the provided documentation (e.g., user guide, API reference)?

- ☐ **1** - Not helpful at all
- ☐ **2** - Somewhat unhelpful
- ☐ **3** - Neutral
- ☐ **4** - Somewhat helpful
- ☐ **5** - Very helpful

12. What is your level of confidence in using Live Neuro after reading the documentation?

- ☐ 1 - Not confident at all
- ☐ 2 - Slightly confident
- ☐ 3 - Neutral
- ☐ 4 - Mostly confident
- ☐ 5 - Fully confident

Section 6: Overall Satisfaction & Suggestions

13. How satisfied are you with Live Neuro overall?

- ☐ 1 - Very dissatisfied
- ☐ 2 - Somewhat dissatisfied
- ☐ 3 - Neutral
- ☐ 4 - Somewhat satisfied
- ☐ 5 - Very satisfied

14. Would you recommend Live Neuro to other neuroscience researchers?

- ☐ **1** - Definitely not
- ☐ **2** - Probably not
- ☐ **3** - Neutral
- ☐ **4** - Probably yes
- ☐ **5** - Definitely yes

15. What features or improvements would you like to see in future versions? (*Open-ended*)

• **Your response:** _____