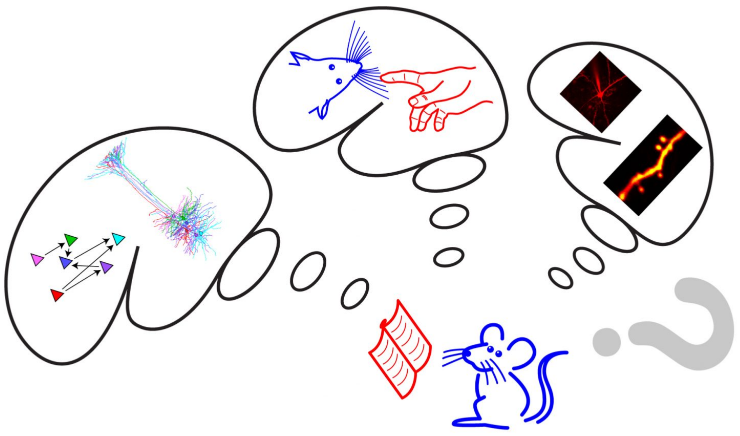
Internship Report

Development of NWB Data Conversion and Visualization Pipelines



Author: Loris Fabbro

Student at EPFL

Supervisor: Dr. Sylvain Crochet

Internship at LSENS – Laboratory of Sensory Processing

EPFL Brain Mind Institute (BMI)

August 2025

Contents

Chapter 1

Introduction

1.1 Context and objectives

This internship was carried out at **LSENS – Laboratory of Sensory Processing**, part of the **EPFL Brain Mind Institute (BMI)**. The lab aims for a causal and mechanistic understanding of sensory perception and associative learning at the level of neurons, synapses, and local circuits in the mammalian brain, with ongoing experiments focusing on reward-based learning and context-dependent sensory perception in mice. I was supervised by **Dr. Sylvain Crochet**, whose teaching and research address the cellular and circuit mechanisms of mammalian brain function.

My main objective was to **standardize heterogeneous neurophysiological datasets** *(Patch-clamp recordings, chronic local field potential (LFP) recordings using high-impedance sharp tungsten microelectrodes, and LFP recordings acquired with NeuroNexus silicon shank probes)* by converting them into the **Neurodata Without Borders (NWB)** format. For some of the formatted datasets, I also developed accompanying visualization and analysis pipelines.

1.2 Work overview

I developed NWB conversion and sometime visualization/analysis tooling for:

- **Published datasets**, including:
  - *Oryshchuk et al., 2024*: Distributed and specific encoding of sensory, motor, and decision information in the mouse neocortex during goal-directed behavior. [2]

  - *Le Merre et al., 2018*: Reward-Based Learning Drives Rapid Sensory Signals in Medial Prefrontal Cortex and Dorsal Hippocampus Necessary for Goal-Directed Behavior. [1]

- **Unpublished datasets**, obtained within the laboratory (LSENS) and likely to be published in the future, currently processed under internal data-handling rules (no external sharing of raw data).

All projects are grouped in a main GitHub repository, which links to each sub-repository and provides a shared `conda` environment and `requirements.txt`.

1.3 Report Layout

This report first introduces the NWB standard and its importance, then presents each developed pipeline with its scientific context, dataset description, processing flow, and usage instructions, before concluding with reflections on the lessons learned and the remaining steps to complete the work.
The purpose of this report is to deliver a clear and accessible guide to the pipelines, offering both the technical details and broader context needed to run or adapt them, while ensuring they can all be executed seamlessly within a single, unified environment.

1.4 LSENS Server Access

The LSENS laboratory has a file server. **Most of the pipelines I developed require specifying the path to this server.** This document explains how to connect and how to correctly write those paths in your scripts and when running pipelines.
The LSENS server is a **network share** (SMB). You connect with your **EPFL credentials**. Once connected, the share appears as a drive (Windows) or a volume (macOS).

By default, access is **read-only**. Write access is only possible if your account is authorized. Firstly, download and install **Cisco AnyConnect** from the EPFL VPN portal. Then, log in with your **Gaspar username** and **password** to establish a secure connection.

Access from macOS (Finder)

1. Open **Finder**.

2. Menu **Go (Aller)** ▷ **Connect to Server...** (shortcut **cmd-K**).
3. Enter the address: ***smb://sv-nas1.rcp.epfl.ch/Petersen-Lab/***
4. Click **Connect**, enter your **username** and **password** (EPFL account).
5. The volume appears in **Locations**.
6. Then navigate to the desired folder, right-click to open a terminal in that location, and type `pwd` to display the full path.

Access from Windows (File Explorer)

1. Open **File Explorer**.
2. In the top menu, select **This PC** ▷ **Map Network Drive**.
3. Choose a drive letter (e.g., `Z:`) and enter the folder path:

   ```
   \\sv-nas1.rcp.epfl.ch\Petersen-Lab
   ```

4. Click **Connect**, enter your **username** and **password** (EPFL account).
5. The network drive appears under **This PC**.

Chapter 2

NWB Pipelines: Conversion and Visualization

2.1 The NWB Data Standard

Neurodata Without Borders (NWB)[3] is an open *data standard* for organizing and sharing neurophysiology data. It provides a unified way to store **data and metadata** from diverse modalities (intracellular/extracellular electrophysiology, optical physiology ...) in a **FAIR** (findable, accessible, interoperable, reusable) format. The standard is built on **HDMF**, with APIs for Python (`PyNWB`) and MATLAB (`MatNWB`), and typically uses **HDF5** as storage (Zarr prototypes exist). NWB promotes **reusability and reproducibility** (rich metadata, validation with NWB Inspector), integrates with many analysis tools (SpikeInterface, CaImAn, suite2p), and supports sharing via archives such as **DANDI**. Using a standardized format ensures comparability across studies, long-term preservation of data, and reduces redundant efforts.

To read and explore NWB files, the recommended tool is **NWB GUIDE**, which provides an intuitive interface for browsing data stored in the NWB format. It simplifies visualization and inspection of datasets without requiring programming skills.

Key resources:

- Official site: `https://www.nwb.org/`
- PyNWB: `https://github.com/NeurodataWithoutBorders/pynwb`
- MatNWB: `https://github.com/NeurodataWithoutBorders/matnwb`

- NWB GUIDE: `https://github.com/NeurodataWithoutBorders/nwb-guide`
- Archive (DANDI): `https://dandiarchive.org/`

2.2 Global Framework

2.2.1 Repository and Environment

All projects are grouped in the main GitHub repository, which links to each sub-repository and provides a single shared `conda` environment named `nwb-lsens`. The environment is created once (`requirements.txt`) and reused across all pipelines to ensure full reproducibility.

2.2.2 Design Principles

All pipelines handle multiple sessions per mouse. In every case, the converter produces one `NWBFile` per session.
Although the NWB standard imposes certain structural constraints, there remains considerable flexibility in how specific datasets can be formatted. For example, some data types were deliberately omitted, while in other cases the choice of how to expose information within the NWB file was left open. There is, of course, no universal solution; however, in consultation with **Sylvain Crochet**, we established a set of conventions to ensure that the formatting is both standardized within the laboratory and logically aligned with the range of experimental paradigms.
A lightweight *session-row* structure (a temporary `pandas` DataFrame) was introduced to collect all data and metadata before writing to NWB. This approach is used in the *Le Merre* and *Banterle* pipelines; *Oryshchuk* adopts a direct file-by-file conversion flow, where the same `.mat` file is reused for each step of the process. This design works but is less optimized compared to the session-row structure, which enforces consistent column naming and simplifies edits to individual NWB modules.
To make maintenance and future changes easier, the codebase is highly modular. Separate modules handle `general` metadata, `acquisition`, `behavior` processing, `intervals` (trials), and either `units` (for extracellular) or `icephys` (for intracellular), plus a dedicated `nwb_saving` module for writing/validation.

Each converter follows a consistent usage pattern: you provide the input locations for raw data and any metadata, and the output directory where the converted NWB files will be saved.

Most converters also accept optional filters to restrict the conversion to specific mice (e.g., `-choice_mouses`).

```
python convert_to_nwb_<dataset>.py <input_folder> <output_folder> [--choice_mouses ...]
```

In the next section, we provide an overview of how the NWB files are organized across the different pipelines

2.2.3 NWB File Description

Scope and conventions. This document consolidates the NWB content produced by three conversion pipelines (P1 - *Oryshchuk et al.*, P2 - *Pierre Le Merre et al.*, P3 - *Lila Banterle et al.*). For each major NWB container, it first states what is common across pipelines, then specifies pipeline-level differences. **WR(+) denotes reward-contingent** sessions; **WR(−) denotes neutral exposure / free-licking sessions.**

1) Overall organization (NWBFile, subject/session, devices, electrodes)

Common across pipelines

Devices are registered in `/general/devices`, and electrodes are defined with anatomical localization; electrophysiology series reference electrodes via `DynamicTableRegion`. The NWB file provides comprehensive subject information in `/general/subject`, including `species`, `strain`, `sex`, `genotype`, `weight`, and `date of birth`, ensuring precise documentation of the biological context of the data. In addition, session-level metadata in `OVERVIEW` are incorporated whenever available, including `experiment_description` (which itself contains session-specific information), `experimenter`, `institution`, `keywords`, `lab`, `notes`, `pharmacology`, `protocol`, `related_publications`, `session_id`, `slices`, `source_script`, `stimulus`, `surgery`, and `virus`. This information facilitates reproducibility, provenance tracking, and integration with other datasets.

Pipeline-specifics

- **P1.** Device inventory includes the **NeuroNexus A1x32 probe**, the **Blackrock headstage**, and the **high-speed camera** used for videography. The electrode table `/general/extracellular-ephys/electrodes` records detailed metadata for each contact, including its Allen CCF coordinates, **shank membership**, and associated brain region label.

- **P2.** Devices include **Amplifier LFP**, **Tungsten Microelectrodes**, **Digitizer**. The electrode table `/general/extracellular-ephys/electrodes` records detailed metadata for each contact, including stereotaxic coordinates in mm relative to *Paxinos/Franklin*, and associated brain region label. Some subject specific fields are incomplete due to the historical nature of the data; however have been incorporated whenever available.
- **P3.** Devices include **Amplifier LFP**, **Patch-clamp Microelectrodes**, **Digitizer**. Focused on **intracellular** recordings: `/general/intracellular-ephys` defines the electrode. The file aggregates multiple **sweeps** from the same neuron:
  - `IntracellularRecordingsTable`: stores electrode, stimulus, and response information for each recording.
  - `SimultaneousRecordingsTable`: groups concurrent recordings.

  Each session is converted into a single NWB file, representing the recordings from one neuron. The electrode corresponds to a **whole-cell patch pipette** (4–7 MΩ, DC current-clamp) located in the **mPFC**.

2) Acquisition (electrophysiology time series)

Common across pipelines

Electrophysiology data are stored in `/acquisition` that reference electrodes in the table; series naming is explicit.

Pipeline-specifics

- **P1.** One `ElectricalSeries` named `ElectricalSeries_LFP` at 2 kHz, referencing selected electrodes via `DynamicTableRegion`.
- **P2.** Same as P1: `ElectricalSeries_LFP` at 2 kHz with `DynamicTableRegion` referencing.
- **P3.** Intracellular sweeps instead of LFP:
  - `CurrentClampSeries`: `Membrane_potential_swXXX` (20 kHz, units V).
  - `CurrentClampStimulusSeries`: `Current_Monitor_swXXX` (20 kHz, units A).

  The same patch electrode is used across sweeps within a session.

3) Processing / behavior module

Common across pipelines

A `behavior` processing module in `/processing/behavior/` typically contains:

- `BehavioralEvents`:
  - `ReactionTimes`: timestamps of the reaction derived from `PiezoLickSignal`;
  - `ResponseType`: timestamps for response class (0=MISS, 1=HIT, 2=CR, 3=FA);
  - `StimFlags`: timestamps encoding whisker stimulation amplitude;
  - `TrialOnsets`: start time of each trial;
  - per-class flags: `correct_rejection_trial`, `false_alarm_trial`, `whisker_hit_trial`, `whisker_miss_trial`.
- `BehavioralTimeSeries`: continuous traces (e.g., whisker angle, lick sensors, EMG when available).

Pipeline-specifics

- **P1.**

  WR(+): `EngagedTrials` (behavioral engagement), `VideoOnsets` (coincident with `TrialOnsets` or $\sim$1 s earlier), `jaw_dlc_licks` (jaw-movement onsets per trial).

  WR(-): some WR(+) events are absent, notably `TrialOnsets` and `ReactionTimes`; additional variables include `Valve_Ind_Assosiation` (manual vs. automatic valve activation) and `Valve_Ind_MouseTriggered` (valve activation by the mouse).

  BehavioralTimeSeries : `JawTrace`, `TongueTrace`, `NoseTopTrace`, `NoseSideTrace`, `WhiskerAngle`, `PiezoLickSignal` (typically only in WR(+)).

- **P2.**

  WR(-): an additional time series is added to represent the timing of reward delivery.

  BehavioralTimeSeries (when present): stores continuous trace `ElectricalSeries_EMG` and `ElectricalSeries_PiezoLickSignal` (rate 2000 Hz, units V).

- **P3.** Additional sweep alignment markers `sweep_start` and `sweep_stop` (multiple trials per sweep).

  WR(–): `reward_onset` timestamps encode reward delivery times.

  BehavioralTimeSeries (when present): `WhiskerAngle` at 200 Hz and `PiezoLickSignal` at 20 kHz.

4) Intervals

Common across pipelines

A session-level `intervals` table is defined per session; each row corresponds to one trial. Columns standardize stimulus timing/amplitude, response windows, outcomes, and lick-related measures:

- `trial_type`: whisker / no_whisker
- `whisker_stim`: 1 if whisker stimulus delivered, else 0
- `whisker_stim_amplitude`
- `whisker_stim_duration` (ms)
- `reward_available`: 1 for WR(+), 0 for WR(–)
- `perf`: 0 = whisker miss; 1 = whisker hit; 2 = correct rejection; 3 = false alarm
- `no_stim`: 1 if no stimulus delivered, else 0
- `no_stim_time`: defined if `no_stim`=1
- `response_window_start_time`
- `response_window_stop_time`
- `lick_flag`: 1 if lick detected, else 0

Pipeline-specifics

- **P1.** There is no dedicated column `whisker_stim_time` (s) to mark the stimulation time, since all stimulations are aligned to the `TrialOnsets` timestamps. We record `lick_time` as the lick time within the response window.

WR(+): store `jaw_dlc_licks` (DLC-derived jaw movements per trial).

WR(-): rows correspond to *stimulus events* but reflect a free-licking design: rewards are not contingent on the stimulus and we keep performance labels (hit/miss).

- **P2.** `whisker_stim_time` (s) is added to mark the stimulation time, and we keep `lick_time` as the within-window lick time.
- **P3.** Several trials can be contained within a single sweep. We store `whisker_stim_time` (s), `whisker_stim_time_relative` (s; relative to sweep start), `whisker_stim_type`, and `lick_threshold`. We also include `Sweep_Start_time`, `Sweep_Stop_time`, `Sweep_ID`, `Session_Number`, `Reward_time` (s), and `Reward_flag` (1 if reward, else 0).

5) Units table with spiking

Common across pipelines

Where single-unit activity is available, a table in `/units` stores **spike times** and **associated metadata**.

Pipeline-specifics

- **P1.** Rich single-unit table including: cluster ID, main channel, depth, average firing rate, waveform duration, refractory-period violations, isolation quality, neuron type (e.g., regular vs. (RSU) fast-spiking (FSU)), precise electrode location, and Allen CCF coordinates & region labels.
- **P2.** Units table is not described.
- **P3.** Poor single-unit table including: cell identity, anatomical target area (e.g., mPFC), recording depth, and neuron type (often unspecified)."

If you are not familiar with the dataset content, it is recommended to consult the original article and, as noted in the NWB section, open a file with NWB Guide to better understand its structure.

2.2.4 Execution Time and Quality Control

The execution time of the pipelines depends strongly on computing power and, in particular, on network bandwidth (Ethernet is strongly recommended over Wi-Fi). On EPFL workstations with a stable Ethernet connection, the approximate runtimes are as follows:

- **Oryshchuk et al., 2024:** The pipeline runs in about one hour in total. WR(-) sessions typically require 15–20 minutes, while WR(+) sessions take 30–40 minutes. As previously mentioned, it is not possible to select individual mice directly; the pipeline processes all available sessions. This limitation can be circumvented by adjusting the input path to a folder containing only the desired mouse.
- **Le Merre et al., 2018 — Chronic LFP:** This is the most time-consuming pipeline, as it includes a large number of sessions and requires loading two separate `.mat` files per mouse (with roughly one hundred mice in the dataset). A full conversion across all mice takes around 3 hours.
- **Lila Banterle et al., unpublished:** This is the fastest pipeline, completing in approximately 10–15 minutes for all mice. The reduced runtime reflects the nature of the dataset, which contains intracellular patch-clamp recordings rather than large LFP arrays.

Some pipelines may fail to execute fully due to server accessibility or memory (RAM) limitations. In such cases, it is advisable to process the data in **batches of mice**, iteratively repeating the procedure until all subjects have been successfully covered.

All outputs were validated with `PyNWB` (`pynwb.validate`), and no validation errors were encountered across the tested datasets. This indicates that the converters are robust and consistently generate valid NWB files.
Nevertheless, it is important to keep in mind that some legacy data contain irregularities. For example, certain mice may have non-existent dates in their metadata, or in the *Banterle* dataset some sweeps lack a defined start time or include repeated rows in the performance matrix. The codebase has therefore been designed to handle such cases gracefully and can be easily adapted (e.g., by relaxing a field, cleaning inconsistent entries) without the need to rewrite entire modules.

2.3 Oryshchuk et al., 2024 – Workflow

This pipeline is available in the following repository: Github

Scientific Context

Original data format & location.

Raw sessions are provided as MATLAB `.mat` files (one file per session) together with a global CSV (`Subject_Session_Selection.csv`) storing per-session metadata (e.g., session type, start time, mouse info). Please refer to the *LSENS Server Access* section for connection instructions. The dataset itself is located in the following directory:
macOS: $\mapsto$ `/Volumes/Petersen-Lab/analysis/Sylvain_Crochet/DATA_REPOSITORY/Oryshchuk_Spike&LFP_2024`
Windows: $\mapsto$ `//sv-nas1.rcp.epfl.ch/Petersen-Lab/analysis/Sylvain_Crochet/DATA_REPOSITORY/Oryshchuk_Spike&LFP_2024`
The `.mat` files must be retrieved from either the **WR- mice** or **WR+ mice** folders.

Dataset origin.

This pipeline targets the dataset described in Oryshchuk et al., 2024 (see [2]), where mice performed a whisker-based detection task during which high-density extracellular activity was recorded in wS1, tjM1, and mPFC (one or two regions per session). High-speed videography, analyzed with DeepLabCut, captured orofacial kinematics (jaw, tongue, nose, whiskers), while licking was monitored with a piezo sensor. Two behavioral contexts are included: in Whisker Rewarded (WR(+), psychometric, reward contingent) sessions, mice were trained to lick within 1 s after a whisker stimulus (go/no-go), whereas in Whisker Non-Rewarded (WR(-), free-licking, non-contingent) sessions, licking occurred freely and rewards were delivered randomly, with whisker stimuli unrelated to reward availability.

Content to be converted.

Signals: This dataset combines high-density electrophysiology and continuous behavioral traces. LFP arrays were sampled at **2 kHz** across **32/64** channels depending on shank configuration (NeuroNexus A1x32 probes), while spike times were extracted with Kilosort2.5 (`https:`

`//github.com/MouseLand/Kilosort`), originally sampled at **30 kHz**, and subsequently curated with the sortingQuality matrix (`https://github.com/cortex-lab/sortingQuality`). In addition, detailed information about shanks and electrodes are available, including brain region assignment. Example unit metadata include cluster ID, main channel, depth, firing rate, waveform width, refractory-period violations, isolation quality, and anatomical CCF coordinates. Complementing these signals, continuous behavioral traces captured orofacial kinematics (jaw, tongue, nose top/side, whisker angle) aligned to video timing, together with the continuous piezo lick signal.

Events & annotations: For rewarded sessions WR(+), this includes trial onsets, whisker stimulation flags and amplitudes, reaction times, engagement flags, video onsets, DeepLabCut-derived jaw onset flags, response categories. In non-rewarded sessions WR(-), events comprise reward-window onsets, reward valve events with provenance (manual/auto, mouse-triggered), whisker amplitudes, and response flags...

2.3.1 Technical Details

Code organization.

- Main script: `convert_to_nwb_for_AO.py`
- Modules in `converters/`:
  - `Initiation_nwb.py` (config & NWB init)
  - `acquisition_to_nwb.py` (extraction of LFP signals and creation of `ElectricalSeries_LFP`)
  - `general_to_nwb.py` (registration of devices, electrode groups, and electrode metadata)
  - `intervals_to_nwb.py` (trial intervals)
  - `behavior_to_nwb.py` (BehavioralEvents/TimeSeries)
  - `nwb_saving.py` (file writing and validation)

Processing flow

1. **Input:** `convert_data_to_nwb_an(input_folder, output_folder)` enumerates `.mat` files and silences stdout. Each file is read with h5py; if a Data group exists it is used, otherwise

the root. Small fix: when |TrialOnsets_All - VideoOnsets| $< 1e^{-3}$, VideoOnsets is set to TrialOnsets_All.

2. **Session dispatcher:** For each file,
   `convert_data_to_nwb_an_mat(mat_file, output_folder)` determines the session type via `Initiation_nwb.Rewarded_or_not(...)`
   which inspects `Subject_Session_Selection.csv`.

3. **YAML configuration:** `files_to_config_Rewarded(...)` or
   `files_to_config_NonRewarded(...)` writes a temporary YAML that aggregates subject and session metadata. For exemple: WR(+) computes camera_start_delay and per-video duration; WR(-) saves the total video duration. This YAML is consumed at initialization and deleted at the end.

4. **Create NWB files:** metadata are passed to
   `Initiation_nwb.create_nwb_file_an(config_path)` to initialize the NWB file.

5. **General & electrodes:** `acquisition_to_nwb.extract_lfp_signal(data, mat_file)` merges available shanks and returns an array ($T \times n\_channels$) and presence flags for wS1/mPFC/tjM1. `general_to_nwb.add_general_container(...)` declares devices (NeuroNexus probe, Blackrock headstage/DAQ, high-speed camera), creates one or two electrode groups (regions: wS1, mPFC, tjM1), adds CCF ml/ap/dv columns, and builds the electrode table and a DynamicTableRegion.

6. **Acquisition (LFP):** `acquisition_to_nwb.add_lfp_acquisition(...)` stores ElectricalSeries_LFP at 2 kHz, referencing the electrode region. This step is identical for WR(+) and WR(-).

7. **Intervals (trials):**

   - WR(+): `intervals_to_nwb.add_intervals_container_Rewarded(...)` adds rows for both stimulus and no-stim periods with columns for amplitude, onset, 1-ms stimulus duration, response-window bounds (0.05–1 s), outcome (HIT/MISS/CR/FA), lick time, jaw flag, and reward_available=1.
   - WR(-): `...NonRewarded(...)` treats each whisker stimulus as a trial, with amplitude, onset, response window, lick flag/time, performance label (hit/miss), and reward_available=0.

8. **Units (spikes):** `units_to_nwb.add_units_container(...)` creates the units table, loads spikets, computes main_channel (adds +32 for the second shank), and stores rich metadata (cluster id, depth, firing rate, waveform width, RPVs, CCF coordinates/labels, neuron type, isolation metrics). Spike sampling rate is recorded as 30 kHz.

9. **Behavior:**

   - WR(+): `behavior_to_nwb.add_behavior_container_Rewarded(...)` writes BehavioralEvents (TrialOnsets, StimFlags, ReactionTimes, EngagedTrials, VideoOnsets, jaw_dlc_licks, ResponseType, plus derived HIT/MISS/CR/FA flags) and BehavioralTimeSeries (jaw/tongue/nose/whisker traces aligned to video onsets, and PiezoLickSignal).
   - WR(-): `...NonRewarded(...)` writes Reward_Window_onset, Reward_time, Valve_Ind_Assosiation, Valve_Ind_MouseTriggered, StimFlags, and hit/miss flags; continuous traces use VideoFrames_Tms. No PiezoLickSignal is added for WR(-) because lick data are absent from the `.mat` files.

10. **Validation & save.** the NWB file is written to the correct output subfolder (*WR+/* or *WR-/*) and validated with PyNWB. If validation errors occur, the file is deleted and the error is logged.

Main Python libraries. `pynwb`, `h5py`, `numpy`, `pandas`, `tqdm`, `pyyaml`, `python-dateutil`, plus standard libraries.

2.3.2 Usage

Please refer to the pipeline's `README` on GitHub and follow the instructions provided in the sections *Work Environment* and *How to use.*

2.3.3 *Ready Graphical User Interface*

This pipeline is available in the following repository: Github
This module extends an EPFL semester project (Sobhan Nili) to support **NWB-formatted** recordings from [2]. It provides an interactive **Gradio** interface to explore trials and plot **Peri-Stimulus Time Histograms (PSTH)** with rich filters.

My Contribution

- **NWB data integration.**

  I extended the original project to ensure full compatibility with NWB files by developing dedicated loaders that map the *Oryshchuk et al.* schema into a unified `data_struct` through the functions `nwb_to_data_struct_rewarded(...)` and `nwb_to_data_struct_non_rewarded(...)`. Implemented session filtering by condition (WR+ vs. WR−) within `load_all_sessions_merged_Selected_files(...)`.

- **Interactive PSTH GUI.**

  Implementation of a Gradio app with two analysis tabs (WR(+) and WR(–)) that natively reads NWB sessions and renders interactive PSTHs. Users select the **alignment event**; *trial onset*, *jaw_ dlc_ licks*, or *lick* for WR(+); *PiezoLickOnsets* or *Coils_ onset* for WR(–); and **filter trials** by stimulus amplitude and responses types (miss, hit, Cr, Fa). Neuronal subsets are chosen by brain region, CCF acronym, neuron type, and quality thresholds (RPV, ISI, ISO); the PSTH time window and bin size are configurable. The PSTH back end (`plot_custom_psth` for WR(+), `plot_custom_psth_non_rewarded` for WR(–)) bins spikes in seconds, normalizes rates by the number of events and bin width, and optionally exports figures.

Files of interest

- `launcher.py`: path setup, mouse filtering, app start.
- `src/tabs/temp_matching/interface.py` (WR+) and `interface2.py` (WR−): GUI and callbacks.
- `src/tabs/temp_matching/utils.py` and `.../callbacks.py`: plotting and analysis helpers.

Processing flow

1) Launch and selection.

1. User runs `python launcher.py`.
2. The launcher asks for WR(+) and WR(-) directories

   (paths can be remembered in `.launcher_prefs.json`).

3. Optional mouse filter (space-separated IDs) is applied.
4. The launcher writes `src/share.py` with:
    - `WR_PLUS_PATH`, `WR_MINUS_PATH`,
    - `MATCHED_FILES_WR_PLUS`, `MATCHED_FILES_WR_MINUS`.
5. It starts the Gradio multi-tab app. Each tab is rendered only if matching files exist.

2) Data load.

1. For WR(+): `load_all_sessions_merged_Selected_files(MATCHED_FILES_WR_PLUS, True)`.
2. For WR(-): same function with `False`.
3. Each NWB file is parsed into a session dictionary (dates, mouse, events, trials, units) and sessions are appended into `data_struct`.

3) UI initialization.

1. The app computes per-session summaries (neuron count, responses types ...).
2. **Sliders** are initialized and **Filters** are populated from the selected session: amplitudes, results, neuron types, regions, CCF acronyms.

4) Interactive filtering.

- *Changing the session index updates:*
    - The session label (`mouse` and date).
    - Quality sliders (RPV/ISI/ISO), trial-type counters, and available filter choices.
- *Alignment choice toggles the relevant widgets:*
    - WR+: `lick_timestamps` shows lick on/off; `trial_onset` shows (stim amp + result); `jaw_dlc_licks` uses DLC-detected timestamps.
    - WR−: `Coils_onset` reveals amplitude selection; `PiezoLickOnsets` aligns on inferred lick events.
- Region selection filters available CCF acronyms, and CCF selection further restricts neuron types.

5) PSTH computation.

1. The handler (`plot_custom_psth` or `plot_custom_psth_non_rewarded`) builds the neuron mask from:
   - RPV $\leq$ threshold, ISI $\leq$ threshold, ISO $\geq$ threshold.
   - Brain region, CCF acronym, neuron type.
2. It extracts alignment events based on the selected mode and (if needed) filters trials by amplitude and/or result.
3. For each kept unit, spikes are aligned to each event, histogrammed into bins, normalized by (#events $\times$ bin width), and averaged across units.
4. The PSTH curve is plotted with 0 s at the alignment time; figures can be saved on demand.

6) Extra analyses (optional).

- Template distances, clustering (UMAP on pairwise distances), ROC, confusion matrix.
- Continuous perception estimation over time with adjustable binning.

Main Python libraries. `gradio`, `gradio_rangeslider`, `pynwb`, `numpy`, `pandas`, `scipy`, `scikit-learn`, `matplotlib`, `umap-learn`, and standard libraries.

Usage

Please refer to the pipeline's `README` on GitHub and follow the instructions provided in the sections *Work Environment* and *How to use.*

In addition, when opening the NWB files with **NWB Guide**, **peri-stimulus time histograms (PSTHs)** for ***single units*** are readily accessible. These visualizations capture the temporal firing dynamics of neurons relative to specific events (e.g., stimulation amplitude, lick flag...). In contrast, the custom interface extends these functionalities by enabling ***averaging across units*** and providing more advanced filtering options based on quality metrics and brain regions.

Users are therefore free to choose between the *NWB PSTHs* or *the Custom Interface PSTH*, depending on their specific analytical needs.

2.4 Pierre Le Merre et al., 2018 – Workflow *Chronic LFP*

This pipeline is available in the following repository: Github.

Scientific Context

Original data format & location.

MATLAB `.mat` files: (i) a **general** mouse-level file per animal containing session metadata (e.g., session dates), and (ii) multiple **session-specific** `.mat` files providing the raw electrophysiological and behavioral recordings.

macOS: ↦ General – `/Volumes/Petersen-Lab/publications/2018/2018_LeMerre_Neuron/2018_LeMerre_Neuron_data/processed_data` and Sessions – `/Volumes/Petersen-Lab/analysis/Sylvain_Crochet/DATA_REPOSITORY/LeMerre_mPFC_2018/Chronic_LFPs_Preprocessed`

Windows: ↦ General – `//sv-nas1.rcp.epfl.ch/Petersen-Lab/publications/2018/2018_LeMerre_Neuron/2018_LeMerre_Neuron_data/processed_data` and Sessions – `//sv-nas1.rcp.epfl.ch/Petersen-Lab/analysis/Sylvain_Crochet/DATA_REPOSITORY/LeMerre_mPFC_2018/Chronic_LFPs_Preprocessed`

An important consideration is that the general MATLAB file is stored in an older format and can only be read using `scipy.io.loadmat`, whereas the individual session files were updated more recently and therefore support MATLAB v7.3 via `mat73`.

Dataset origin

This subsection documents only the conversion of chronic multisite LFP data from Le Merre et al. (see [1]) to the NWB standard. Chronic multisite LFP recordings in mice performing a whisker sensory detection task (WR(+), "Detection Task") or undergoing Neutral Exposure WR(-)). Target areas include: wS1, wS2, wM1, PtA, mPFC, dCA1 (optionally antM1/EEG/EMG). Electrode implantation used high-impedance tungsten microelectrodes; licking was detected by a piezo film at the reward spout.

Content to be converted.

Signals: LFP arrays per region (PtA, dCA1, mPFC, wM1, wS1, wS2, antM1) sampled at 2000 Hz; optional continuous signals include EMG (2000 Hz), EEG (2000 Hz), and PiezoLickSignal (2000 Hz)."

Events & annotations: The conversion particularly targets the behavioral information and trial structure: onset and offset of each trial, presence or absence of whisker stimulation and its amplitude, licking responses and their timing, as well as trial outcomes (miss, hit, correct rejection, false alarm).
In Neutral Exposure (WR(-)) sessions, the timing of reward delivery is also included.

2.4.1 Technical Details

Code organization.

- Main script: `convert_to_nwb_for_PL.py`
- Modules in `converters/`:
  - `Initiation_nwb.py` (config & NWB init)
  - `acquisition_to_nwb.py` (extraction of LFP signals and creation of `ElectricalSeries_LFP`)
  - `general_to_nwb.py` (registration of devices, electrode groups, and electrode metadata)
  - `intervals_to_nwb.py` (trial intervals)
  - `behavior_to_nwb.py` (BehavioralEvents/TimeSeries)
  - `nwb_saving.py` (file writing and validation)

Processing flow

1. **Input:** the pipeline loads the general `.mat` file for each animal together with the corresponding session-level `.mat` files + optional filter `-choice_mouses`.
2. **Pair discovery:** match each mouse folder in `processed_data` with a session file in `Chronic_LFPs_Preprocessed` (`find_pl_pairs`).
3. **Session initialization and session table construction:** for each mouse, the pipeline iterates over the pairs identified by `find_pl_pairs`. For every match, it generates a mouse-specific DataFrame with `files_to_dataframe()`. Internally, this function builds a dictionary of session attributes (dates, behavioral labels, electrophysiological signals, event markers), which is then transformed into a tabular DataFrame. Each iteration appends multiple rows, one per session.

4. **YAML configuration:** a temporary YAML file is generated using `converters.Initiation_nwb.files_to_config(...)`. It organizes subject/session metadata and experiment parameters before being deleted.

5. **Create NWB file:**
   metadata are passed to `converters.Initiation_nwb.create_nwb_file(...)` to initialize the NWB file.

6. **General & electrodes:**

   - `extract_lfp_signal(csv_data_row)` parses LFP regions (PtA, dCA1, mPFC, wM1, wS1, wS2, antM1), builds a $(T \times N)$ matrix, and returns the matrix + channel labels.
   - `add_general_container(nwb_file, regions)` registers Device objects, creates an ElectrodeGroup ("LFP"), and appends electrodes with stereotaxic coordinates & anatomical location. Returns a `DynamicTableRegion` and channel labels.

   Together, these ensure a 1:1 mapping between matrix columns and electrode indices.

7. **Acquisition:** merge available LFP regions into a $(T \times N)$ array; write `ElectricalSeries_LFP` at 2000 Hz (units V, filter 0.1–1000 Hz).

8. **Intervals (trials):** use `intervals_to_nwb.add_intervals_container(...)` to store trial-level data (onsets, stimuli, licks, outcomes, catch trials) in NWB `TimeIntervals`.

9. **Behavior:** handled by `behavior_to_nwb.add_behavior_container(...)`.

   - Adds `BehavioralEvents`: trial onsets, reaction times, stimulus flags, response types, hit/miss/CR/FA, reward onsets WR(-).
   - Adds `BehavioralTimeSeries`: continuous signals (EMG, Piezo lick traces).

10. **Save & validate:** the NWB file is written to the correct output subfolder (*Detection Task/* or *Neutral Exposition/*) and validated with PyNWB. If validation errors occur, the file is deleted and the error is logged.

2.4.2 Usage

Please refer to the pipeline's `README` on GitHub and follow the instructions provided in the sections *Work Environment* and *How to use.*

2.5 Lila Banterle et al., unpublished – Workflow

This pipeline is available in the following repository: Github.

Scientific Context

Original data format & location.

The primary source is a MATLAB `.mat` file (`mPFC_Preprocessed.mat`) containing electrophysiological recordings along with partial metadata. Additional subject and session attributes not present in the `.mat` file, such as ear tag and reference weights, are provided in a complementary CSV file (`Subject_Session_Selection.csv`). Please refer to the *LSENS Server Access* section for connection instructions. The dataset itself is located in the following directory:
macOS: $\mapsto$ `/Volumes/Petersen-Lab/analysis/Sylvain_Crochet/DATA_REPOSITORY/Banterle_mPFC_Vm_2019/mPFC_Preprocessed.mat`
Windows: $\mapsto$ `//sv-nas1.rcp.epfl.ch/Petersen-Lab/analysis/Sylvain_Crochet/DATA_REPOSITORY/Banterle_mPFC_Vm_2019/mPFC_Preprocessed.mat`

Dataset origin.

Single-neuron intracellular recordings (whole-cell current-clamp) from mouse medial prefrontal cortex (mPFC) during either (i) **Whisker Rewarded WR(+)** sessions, water reward available within 1 s after whisker stimulation on go trials, but not in no-go trials or (ii) **No-Task** sessions with randomly delivered whisker stimuli under head fixation.

Content to be converted.

For each **cell** and its **sweeps**:
Signals: Membrane potential traces sampled at 20 kHz, current monitor signals recorded at 20 kHz to track injected currents, whisker angle measurements acquired at 200 Hz to capture sensory input, and licking activity measured at 20 kHz with a piezo sensor, complemented by a defined threshold for lick detection.
Events & annotations: AP times, reward times, stimulus times (onset, amplitude, duration, type), per-trial behavioral matrix (trial time, stimulation flag, lick indicator, reward flag, and responses types).

2.5.1 Technical Details

Code organization

- Main script: `convert_data_to_nwb_LB.py`.
- Modules `converters/`:
  - `Initiation_nwb.py` (config & NWB init),
  - `intervals_to_nwb.py` (trial/sweep intervals),
  - `acquisition_and_unit_to_nwb.py` (general devices + icephys series + neuron metadata),
  - `behavior_to_nwb.py` (BehavioralEvents/TimeSeries),
  - `nwb_saving.py` (save file).

Processing flow

1. **Input:** path to the `.mat` file + `Subject_Session_Selection.csv` + optional filter `-choice_mouses`.
2. **Loading :** extract mouse/cell metadata, sweeps (Vm, current, whisker angle, APs, licking, stimulation, rewards), sweep timestamps, and behavioral matrix (n×5). Then, **data cleaning** is performed: duplicate trials are removed, all time values are converted into a consistent format (first to `datetime`, then to seconds), robust helper functions (`safe_* and read_*`) are applied to handle missing or malformed entries.
3. **Session initialization and build session row:** For each selected cell ID, the function `to_data_frame` is called. Each session row is constructed from a structured dictionary that stores key–value pairs for all metadata and experimental parameters, together with a **list of sweeps**. Each **sweep** is itself represented as a nested **dictionary**, containing raw signals, spike times, reward events, and detailed **trial dictionary** with stimulus and response annotations etc.
4. **YAML configuration:** A temporary YAML file is generated using `converters.Initiation_nwb.files_to_config(...)` from the session dictionary. It organizes subject and session metadata together in a structured format, and is used by the NWB initialization function before being deleted.

5. **Create NWB file:** Subject and session metadata are provided to `converters.Initiation_nwb.create_nwb_file(...)`, which initializes the NWB file.

6. **Intervals (trials):**
   The function `converters.intervals_to_nwb.add_intervals_container(...)` iterates over `csv_data_row["sweeps"]` to build a trial table by traversing each **sweep's trials dict**.

7. **Acquisition icephys:** Executed via `converters.acquisition.add_to_nwb_acquisition_and_units_containers()`. For each sweep, the function creates a `CurrentClampSeries` (Vm trace) and, when a current monitor signal is available, a `CurrentClampStimulusSeries`. Both are linked to the same icephys electrode and added as an `intracellular_recording` entry. These entries are then grouped into a `simultaneous_recording` block. Finally, spike times detected in the sweeps are aggregated and stored in the **units** table with their associated metadata.

8. **Behavior:** Executed via `converters.behavior_to_nwb.add_behavior_container(...)`. The function iterates through each sweep in `csv_data_row['sweeps']` to collect behavioral markers.

9. **Validation & save:** The pipeline writes the NWB file to the appropriate output subfolder (*WDT/* or *No Task/*) and runs PyNWB validation. If validation errors are detected, the file is deleted and the error is logged.

Main Python libraries. `pynwb`, `h5py`, `numpy`, `pandas`, `tqdm`, `pyyaml`, `python-dateutil`, plus standard libraries.

2.5.2 Usage

Please refer to the pipeline's `README` on GitHub and follow the instructions provided in the sections *Work Environment* and *How to use.*

Chapter 3

Discussion

This internship achieved its main objective: the standardization of heterogeneous neurophysiological datasets into the **NWB format**. Three major pipelines were developed, covering extracellular spikes and LFPs (*Oryshchuk 2024*), chronic LFP recordings (*Le Merre 2018*), and intracellular patch-clamp recordings (*Banterle*).But, some practical design choices, remain open for refinement. In addition to the conversion workflows, I also implemented a graphical user interface with Gradio, allowing interactive visualization of peri-stimulus time histograms. All pipelines were integrated into a unified `conda` environment and a central repository, ensuring reproducibility and easier maintenance. Systematic validation with PyNWB confirmed the robustness of the outputs, even when dealing with legacy data irregularities. This work therefore provides a strong backbone for future data reuse and sharing within the laboratory and, eventually, for public deposition on DANDI.

An important next step will be to ensure that the datasets fully align with the evolving standards and best practices of the neuroscience community, thereby facilitating their reuse by other laboratories. While the code developed is functional, it is unlikely to be flawless, and I sincerely hope that no errors were introduced in the conversion process.
The pipelines could be further strengthened with additional safeguards and exception handling to better cover edge cases, which is particularly important given the scale of the datasets (over 100 GB of raw data and metadata). As with any complex workflow, implementing the pipelines required making design choices that inevitably involve trade-offs. While there will always exist alternative implementations that are faster, more elegant, or more readable, I strived to deliver

code that remains functional, clear, and reliable for both current use and future needs of the laboratory.

Beyond the technical outcomes, this project allowed me to acquire valuable skills. I gained practical expertise in PyNWB, HDF5 data handling, modular pipeline design, and lightweight graphical interfaces, as well as experience in validation, documentation, and reproducible workflows. Scientifically, I deepened my understanding of whisker-based behavioral paradigms and how neuronal signals relate to task engagement and reward, while also applying FAIR data principles to real experimental datasets.

In summary, this internship provided a **standardized and validated NWB framework** for some LSENS data. Recordings are now organized in a format that is sustainable, interoperable, and ready for further analysis and dissemination. With the proposed extensions, these tools can evolve into a lasting platform that will benefit both the laboratory and the wider neuroscience community.

Acknowledgements

I would like to sincerely thank **Dr. Sylvain Crochet** for his supervision and guidance throughout this project, as well as the members of **LSENS** for their valuable feedback and support.

Bibliography

[1] P. Le Merre et al. “Reward-Based Learning Drives Rapid Sensory Signals in Medial Prefrontal Cortex and Dorsal Hippocampus Necessary for Goal-Directed Behavior”. In: *Neuron* 97.1 (2018). Epub 2017 Dec 14, 83–91.e5. DOI: `10.1016/j.neuron.2017.11.031`.

[2] Anastasiia Oryshchuk et al. “Distributed and specific encoding of sensory, motor, and decision information in the mouse neocortex during goal-directed behavior”. In: *Cell Reports* 43.1 (2024), p. 113618. ISSN: 2211-1247. DOI: `https://doi.org/10.1016/j.celrep.2023.113618`. URL: `https://www.sciencedirect.com/science/article/pii/S2211124723016303`.

[3] Oliver Rübel et al. “The Neurodata Without Borders ecosystem for neurophysiological data science”. In: *eLife* 11 (2022). Ed. by Laura L Colgin and Shantanu P Jadhav, e78362. ISSN: 2050-084X. DOI: `10.7554/eLife.78362`. URL: `https://doi.org/10.7554/eLife.78362`.