­­­­­­­­­­Paper Goals / *Target Audience*:

* Overview of software suite for *Users*
* Peer reviewed (legitimized) reference for *Citations*
* Detailed account of technical challenges and solutions for *RSE* *peers*

1. INTRODUCTION
   1. NWB and DANDI address a critical need for data standardization and sharing in the neurophysiology community.
      1. NWB and DANDI facilitate data sharing for reanalysis among neurophysiologists.
      2. NWB and DANDI streamline data for software tool developers.
      3. NWB and DANDI effectively archive data by storing contextual information in metadata alongside all objects.
   2. Despite the myriad benefits, many labs struggle to adopt the data standard because of high barriers to entry including
      1. Learning unfamiliar data organization i.e. how NWB works
      2. Learning how to interact with NWB using data APIs (pyNWB or MatNWB)
         1. Note: this step can be difficult without solid programming experience
      3. Converting existing lab data *post hoc*
         1. Note: this step can end up being a lot of work due to a backlog of data and often individual lab members lack strong incentives to accomplish this
      4. Building conversion pipelines for future data
      5. Educating collaborators unfamiliar with the data format
   3. To address these common barriers we have developed a software suite that automates and simplifies the conversion to NWB with 3 core tools:
      1. nwbinspector – A CLI tool that ensures the validity of NWB files so that they can be reliably uploaded to sharing sites such as DANDI
      2. neuroconv – A Python package for converting neurophysiology data in a variety of proprietary formats to nwb
         1. neuroconv also facilitates the development of automatic conversion pipelines
      3. NWB GUIDE - an interactive, user-friendly application for converting data to the NWB standard for no-code users.
         1. Powered by neuroconv.
2. NWB Inspector
   1. Problem Statement: It can be time-consuming and error-prone to manually inspect all of the NWB files in a dataset, especially as datasets grow very large (terabytes). New users may also be unfamiliar with NWB best practices, further hampering manual inspection.
   2. NWB Inspector solves this problem by automatically inspecting NWB files, providing three tiers of warnings/errors:
      1. Best Practice Violation: A clear violation of known best practices, for example, negative spike times.
      2. Critical: A potentially critical mistake, such as zero-rate timestamps.
      3. Best Practice Suggestion: A suggestion for an improvement to the NWB file, such as adding an informative description.
   3. Implementation Details:
      1. CLI
      2. Python API
      3. Extensions and Configurations, for example, DANDI.
3. Neuroconv
   1. Problem/Solution Statements
      1. Problem: It can be time – consuming and error – prone to manually convert each neurophysiology data set to NWB using pynwb alone, especially for new users unfamiliar with NWB. It can also be difficult to create automatic lab-specific conversion pipelines due to data complexity, heterogeneity, and volume.
      2. Solution: Neuroconv addresses these problems by providing turn key conversion of 41 popular neurophysiology data formats and a programmatic scaffolding for custom conversions.
   2. Automatic conversion of common formats
      1. Challenges
         1. Variety: Neurophysiology data spans multiple modalities including voltage recording, spike sorting, optical imaging, cell segmentation, and behavior. Each modality is represented by various different software – dependent formats with little standardization.
         2. Proprietary software: frequently neurophysiology data is acquired/generated using proprietary software, which introduces a host of additional challenges including acquiring example data for testing and licensing issues with running proprietary data readers.
         3. Dynamic needs: many common formats are changing rapidly to accommodate new developments in the field such as 3-D imaging for optical physiology. This leads to increased maintenance cost for neuroconv to support all of its formats.
      2. Solutions
         1. To address the variety of data modalities we depend on different subpackages for each modality. For example, neo for electrical physiology and roiextractors for optical physiology. Each modality – specific sub package defines a common API and reads all of the different data formats into that API. Then, neuroconv converts the standardized API into nwb. This way the sub packages handle all of the different formats, so neuroconv can focus on converting standardized data into nwb.
         2. To do: how do we handle proprietary software concerns?
         3. To keep up with that dynamic needs of the field, we
            1. Maintain an active presence in the community ex. developer days, caiman workshop, etc.
            2. Participate in the development of NWB extensions, which helps keep us abreast of the latest developments ex. ndx-holographic-stimulation.
            3. Employ continuous integration testing, which helps catch breaking changes as they occur.
      3. Implementation details
         1. Example usage
         2. Metadata yaml
         3. Interface combinations
   3. Programmatic scaffolding for custom conversions
      1. Challenges:
         1. Data complexity: neurophysiology experiments typically involve multiple different streams of Data, so custom conversions need to handle all the relevant types of data for a given lab.
         2. Data heterogeneity: different experiments even from the same lab often tend to have small differences that make their data very heterogeneous
         3. Data volume: neurophysiology experiments are growing increasingly large with up to TBs of data per experiment.
      2. Solutions:
         1. Modular: the building blocks of our conversions are data interface classes. Each data interface is responsible for a specific format of data, and contains methods to read data and meta-data from that format and write it to nwb. Since each data interface is designed as a completely independent unit, they are easy to mix and match together with custom data interfaces.
         2. Extensible: all built in Data interfaces inherit from an abstract base class called base data interface. New custom data interfaces can simply inherit from the base data interface, making the conversion easily extensible.
         3. Automatic: conversions often require multiple different data streams each with its own corresponding Data interface. The NWB converter class allows you to combine multiple data interface objects into a single conversion, and provides methods to aggregate and synchronize data across multiple sources. Due to the standardized structure of data interfaces this is done with minimal boilerplate. Since lab conversions often have a very similar structure, we also provide a cookie cutter github repository to automatically generates full conversion repository, streamlining the conversion development process.
         4. State of the art chunking and compression for large data volumes
   4. High quality documentation makes adoption easy
      1. User guide
      2. Conversion gallery
      3. Project catalog
      4. Doc string standardization
   5. On going challenges
      1. Temporal alignment
      2. ?
4. NWB guide
   1. Problem/solution statements:
      1. Problem: converting a lab to the NWB Data standard requires at least some proficiency with either python or matlab. However, not all neurophysiologists are proficient programmers or have easy access to proficient programmers. Even if a lab has some programming experience, pynwb, matnwb, and neuroconv can be intimidating and time – consuming libraries to learn, which can be an additional burden to a subset of lab members who do have some programming experience.
      2. Solution: NWB Guide solves these problems by providing a user friendly GUI to convert data to nwb. This tool allows users to convert Data without writing a single line of code.
   2. Neuroconv back end
      1. Challenges: ???
      2. Solutions: ???
   3. Electron front end?
      1. Challenges: ???
      2. Solutions: ???
   4. Adoption
      1. SfN
   5. Limitations
      1. Less flexible
5. Discussion
   1. High-level summary
   2. Comparison to similar Data conversion packages
   3. Next steps