Neuron

Supplemental Information

Neurodata Without Borders: Creating a Common Data Format for Neurophysiology

Jeffery L. Teeters, Keith Godfrey, Rob Young, Chinh Dang, Claudia Friedsam, Barry Wark, Hiroki Asari, Simon Peron, Nuo Li, Adrien Peyrache, Gennady Denisov, Joshua H. Siegle, Shawn R. Olsen, Christopher Martin, Miyoung Chun, Shreejoy Tripathy, Timothy J. Blanche, Kenneth Harris, György Buzsáki, Christof Koch, Markus Meister, Karel Svoboda, and Friedrich T. Sommer

SUPPLEMENT

A. Releases of the NWB:Neurophysiology pilot project

The July 2015 release of the NWB initiative includes code and documentation for the specification language, APIs, and tools. These are available at: https://github.com/NeurodataWithoutBorders.

A1. Format Specifications

Description of features of the NWB format and its specification language.
https://github.com/NeurodataWithoutBorders/specification

- **Format Specification** (PDF) - Specification of the format in English text.
- **NWB Schema** - JSON file that uses specification language to define the NWB format. (Created from nwb_core.py used with the Python API).
- **Specification Language Documentation** (PDF) –Detailed reference for the NWB specification language.

A2. Application Programming Interfaces (API)

Python and MATLAB application programming interfaces (API) are provided for data publishers. Both implementations refer to a schema file written in the specification language.

- **Write API** (Python) - Github repository for the Python Write API's source code. https://github.com/NeurodataWithoutBorders/api-python
- **Write API (MATLAB) -** Github repository for the MATLAB Write API's source code and unit tests. https://github.com/NeurodataWithoutBorders/api-matlab

A3. NWB Tools

The following is a list of tools that are available to work with the NWB format.

- **File Diff Tool** (Python) - Script that compares two files in the NWB format. https://github.com/NeurodataWithoutBorders/diff
- **Transform MATLAB Data File to HDF5** (Python) - Recursively transforms data in the MATLAB format to a more easily processed HDF5 format. https://github.com/NeurodataWithoutBorders/mat2h5

A4. Additional Allen Institute NWB Resources

The Allen Institute for Brain Science provides a software development kit for the Allen Cell Types Database and a write API for the NWB format written in Python. The API contains a few features not yet included in the official version and is not currently based on the specification language.

- **Allen SDK** (Python) - Software development kit for reading experimental data and running models from The Allen Cell Types Database. http://alleninstitute.github.io/AllenSDK/
- **Python Write API for the NWB Format** (Python) - Object oriented API for writing to a NWB file. https://github.com/AllenInstitute/nwb-api

A5. Exemplar Datasets

For all of the use cases, sample datasets in the new format are available at CRCNS.org (https://crcns.org/NWB/exemplar-data-sets). In the following list, the id in parenthesis (e.g. "hc-3") indicates the name of the data set at CRCNS.org containing the original data, i.e. before conversion to the NWB format, and the link is to example files of the data in the NWB format.

- **Rat Hippocampus (hc-3)** – Multi-unit recordings from different rat hippocampal regions while the animals were performing several behavioral tasks. Data from the Buzsáki lab. https://portal.nersc.gov/project/crcns/download/nwb-1/hc-3
- **Mouse Visual Cortex (pvc-6)** – In vitro intracellular recording and staining of a single neuron in the visual cortex of a mouse. Data from the Allen Institute for Brain Science. https://portal.nersc.gov/project/crcns/download/nwb-1/allenInst
- **Mouse Premotor Cortex (alm-1)** – Extracellular recordings from neurons in the anterior lateral

motor cortex of adult mice performing a tactile decision behavior. Data from the Svoboda lab. https://portal.nersc.gov/project/crcns/download/nwb-1/alm-1

- **Mouse Vibrissal S1 (ssc-1)** – Calcium imaging from vibrissal somato-sensory cortex 1 in mice performing a pole localization task. Data from the Svoboda lab. https://portal.nersc.gov/project/crcns/download/nwb-1/ssc-1
- **Mouse Retina (ret-1)** – Multi-electrode recording on ex-vivo retina, with visual stimulus. Data from the Meister lab. https://portal.nersc.gov/project/crcns/download/nwb-1/ret-1

B. The use cases in the NWB pilot project

The use cases considered in the project included rodent experiments with different behavioral paradigms and recording techniques from published studies. The studies from the Buzsáki lab included multi-electrode recordings in hippocampus and entorhinal cortex in rats exploring mazes (Pastalkova et al., 2008; Mizuseki et al. 2009; 2011; 2014; Mizuseki & Buzsáki, 2013). The studies from the Svoboda lab included extracellular recordings from ALM neurons of adult mice performing a tactile decision behavior (Li et al., 2015), and calcium imaging data from vibrissal S1 in mice performing a pole localization task (Peron et al., 2015). The studies from the Meister lab included single-unit neural responses recorded from isolated retina from mice using a 61-electrode array in response to various visual stimuli (Lefebvre et al., 2008; Zhang et al., 2014). The use cases from the Allen Institute for Brain Science included slice physiology using whole-cell recordings (Berg, 2014), and in-vivo multi-electrode recordings during visual stimulation in cat visual cortex (Blanche, 2005). A list with more detailed information about each of the use cases is at http://crcns.org/NWB/Data_sets.

C. Examples of how data structures are defined in the specification language

In the "what" document the information required to describe a certain type of experiment was formulated in a pseudocode, intended to be independent of any particular storage method. A few examples from the "what" document are given here along with corresponding expressions in the specification language.

C1. Simple pieces of metadata

The following metadata in the "what" document describe properties of the experimental animal or subject. These descriptions consisted of a key-value pair:

```
Species - Text (use biology-wide standard)
Genotype - Text (use biology-wide standard)
Sex - Text (M/F)
```

They are expressed in the specification file (nwb_core.py) by:

```
 "subject/": {
        "species": { "data_type": "text",
               "description": "Species of subject"},
        "genotype": { "data_type": "text",
               "description": "Genotype of subject"},
        "sex": {
               "data_type": "text",
               "description": "Gender of subject"}, ...
    },
```

C2. Array structure containing data and related metadata

In the "what" document, data recorded from electrodes and associated metadata is described by the following pseudocode. The "for statement" (similar to that statement in programming languages) indicates that descriptions inside the loop-block apply to each instance in a collection of data items of the same type:

```
# Recordings from electrodes
For p in probes
    probe source - text (manufacturer, part number)
    probe location
    For g in channel groups
        Estimated tip location
        ground - text
        For c in channels
            Channel location - numeric: x, y, z (µm)
            Raw data array index
            Impedance - numeric, Ohm
```

The section in the pseudocode describing the raw data, electrode locations and impedances are expressed in the specification file (nwb_core.py) by:

```
"<ElectricalSeries>/": {
       "description": "Acquired voltage data from extracellular recordings.",
       "merge": ["<timestamps>/"],
       "attributes": {"ancestry": "TimeSeries, ElectricalSeries" }},
       "data": {
              "description": "Recorded voltage data",
              "dimensions": ["timeIndex", "channelIndex"], # specifies 2-d array
              "data_type": "number",
              "unit": "volt"},
       "electrode_idx": {
              "description": "Indices to electrodes in electrode_map",
              "dimensions": ["channelIndex"],
              "data_type": "int",
              "references":
   "/general/extracellular_ephys/electrode_map.electrode_number"},
       },
"extracellular_ephys/": {
        "electrode_map": {
               "description": "Physical location of electrode, x,y,z in meters",
               "dimensions": ["electrode_number","xyz"], # specifies 2-D array
               "data_type": "number",
               "xyz" : { # definition of dimension xyz
                      "type": "struct",
                      "components": [
                             { "alias": "x", "unit": "meter" },
                             { "alias": "y", "unit": "meter" },
                             { "alias": "z", "unit": "meter" } ] }
        },
       "impedance": {
               "description": "Impedance of electrodes in electrode_map",
               "dimensions": ["electrode_number"], # specifies 1-D array
               "data_type": "text"
        },
}
```

D. Writing and reading NWB files

This section provides some additional information and code for how to create and read NWB files.

D1. Writing NWB files

The Python and MATLAB write APIs for the NWB format contain two main functions: "`make_group`" which makes groups in the HDF5 file and "`set_dataset`" which create HDF5 datasets. In addition there is a function "`set_attr`" for setting attributes. The specific arguments in the calls to these functions (i.e. which groups, datasets and attributes can be created) are determined by the definition of the NWB format written in the specification language. The following example calls demonstrate the usage of the NWB APIs. The examples are taken from scripts creating the NWB file for the "alm-1" dataset at CRCNS.org, contributed by the Svoboda lab. They illustrate how to write text metadata (the species) and a time series recording of numeric data. The numeric data is called "`lick_trace`" and contains the signal from a photodiode, which detects licking movements. The codes for the Python and MATLAB APIs are very similar.

Python
```
# open NWB file for writing
import nwb_file
f = nwb_file.nwb_file(output_file_name, start_time)

# Store species metadata
s = f.make_group("subject")
s.set_dataset("species", "Mus musculus")

# Create group for lick_trace timeseries
g = f.make_group("<TimeSeries>", "lick_trace",
  path="/acquisition/timeseries")

# Store datasets for timeseries
d = g.set_dataset("data", lick_trace_data)
t1 = g.set_dataset("timestamps", lick_trace_timestamps)
g.set_dataset("num_samples", len(lick_trace_data))
```

MATLAB
```
% open file for writing
f = nwb_file(output_file_name, start_time);

% Store species metadata
s = f.make_group("subject");
s.set_dataset("species", "Mus musculu");

# Store datasets for lick_trace timeseries
g = f.make_group("<TimeSeries>", "lick_trace", "path", ...
  "/acquisition/timeseries");
g.set_dataset("data", lick_trace);
t1 = g.set_dataset("timestamps", timestamps');
g.set_dataset('num_samples', int64(length(lick_trace)));
```

D2. Reading NWB files

Currently the software for the NWB format does not include a read API. A read API is not necessarily required because the format was designed so that data can be easily read using direct calls to HDF5 library functions. Examples of these calls are given below for both Python and MATLAB.

Python

The following example is taken from a script reading the hc-3 dataset at CRCNS.org, contributed by the Buzsáki lab. The script plots data from extracellular recordings in hippocampal areas of rats while they are moving in a confined space. The resulting plots show the position of the animal when the specific neuron generated a spike, color-coded by the approximate direction of movement at the time of the spike. To load the data for plotting for a particular unit, the following code is used:

```
import h5py
# open NWB file
infile = "ec013.156.nwb"
f = h5py.File(infile, "r")
```

```
# function to find all interfaces of a specific type
def get_interfaces(f, itype):
       ilist = []
       proc = f["processing"]
       for k in proc.keys():
       if itype in proc[k].attrs["interfaces"]:
       ilist.append(k)
       return ilist

# find place modules
ilist = get_interfaces(f, "Position")
if not ilist:
  print "Error -- cannot find Position data"
  sys.exit(1)
# use the first; make path to position data
place_mod = "processing/" + ilist[0] + "/Position/position"

# load position time series
pos = f[place_mod]["data"].value
# load times corresponding to each position
post = f[place_mod]["timestamps"].value

# find modules containing UnitTimes interfaces
unit_mods = get_interfaces(f, "UnitTimes")

# loop through all modules containing unit times
for unit_mod in unit_mods:
  # group containing unit times
  grp = f["processing"][unit_mod]["UnitTimes"]

  # loop through each unit
  unit_list = grp["unit_list"].value
  for unit in unit_list:

    # Load spike times for unit
    unit_t = grp[unit]["times"]

    # Now can make plot for unit
```

Once the data is loaded, the plot shown below can be generated. (The code to generate the plot is not shown).

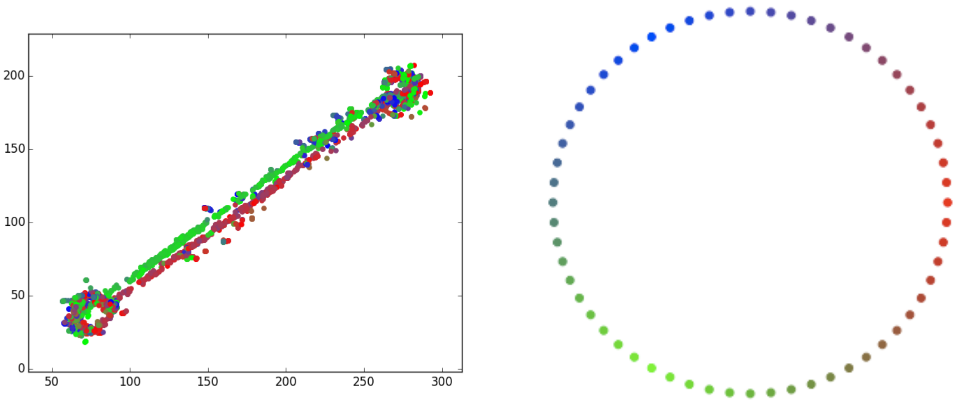

Figure S1: (Left) Plots of spikes from one neuron displayed at the point in the environment that the animal traversed when the spike occurred. Colors indicate the approximate direction of movement at the time of the spike using the color code shown on the right, with movement from center outward. (Data is from hc-3 dataset in CRCNS.org, which was contributed by the Buzsáki lab).

MATLAB

The following sample code is from a script for the alm-1 dataset at CRCNS.org, contributed by the Svoboda lab. The script creates plots showing the response during multiple trials recorded from anterior lateral motor cortex (ALM) neurons of adult mice, for two behaviors (a left or right movement) and also a histogram of the responses. To load the data for plotting a particular unit, the following code is used:

```matlab
% open NWB file
infile = 'NL_example20140905_ANM219037_20131117.nwb';
fid = H5F.open(infile);

% get list of units (this depends on the name of the module being "Units")
unitList = h5read(infile, '/processing/Units/UnitTimes/unit_list');
% get number of trials
epochGroup = H5G.open(fid, '/epochs');
numTrials = H5G.get_info(epochGroup).nlinks;

% get properties of each trial
HitR = zeros(1, numTrials);
HitL = zeros(1, numTrials);
t_TrialStart = zeros(numTrials,1);
for i = 1:numTrials
        trialPath = sprintf('/epochs/Trial_%03i', i);
        trialTypes = h5read(infile, strcat(trialPath, '/tags'));
        trialTypes = deblank(trialTypes);
        if any(ismember(trialTypes, 'HitR'))
                HitR(1,i) = 1;
        end
        if any(ismember(trialTypes, 'HitL'))
                HitL(1,i) = 1;
        end
        startTime = h5read(borgFilePath, strcat(trialPath,'/start_time'));
        t_TrialStart(i,1) = startTime;
end

% Loop through all units (to generate plot for each one)
for i_unit = 1:numel(unitList)
        currUnit = unitList{i_unit}

        % Load unit spike times
        dataPath = strcat('/processing/Units/UnitTimes/', currUnit);
        times = h5read(infile, strcat(dataPath, '/times'));

        % Make plot for unit (code not shown)
end
```

The plot of trial responses and histograms for one unit is shown in Figure S2.

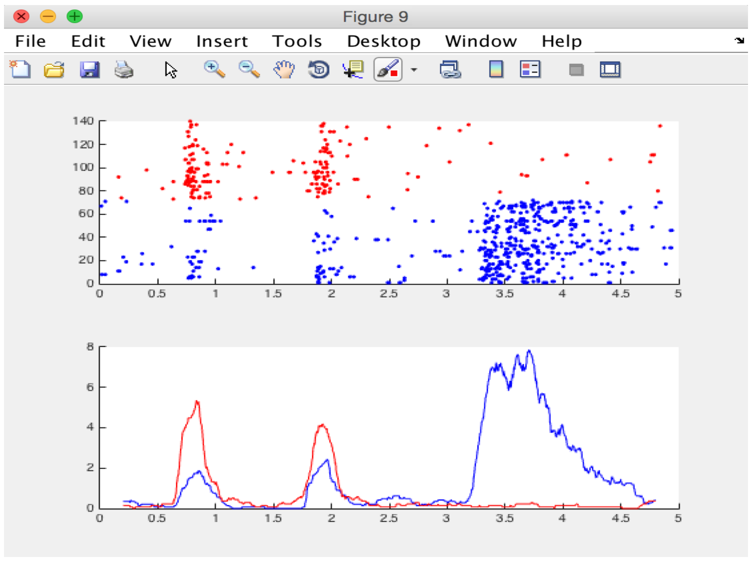

Figure S2: Trial raster plots (upper plot) and PSTHs (lower plot) for one neuron (red = trials with left movement, blue = trials with right movement). From alm-1 dataset in CRCNS.org, contributed by Svoboda lab.

SUPPLEMENTAL REFERENCES

Berg, J. (2014) In vitro whole-cell patch clamp recordings from visual cortex neurons in the adult mouse. CRCNS.org. http://dx.doi.org/10.6080/K0H12ZXD

Blanche T. J., Spacek M. A., Hetke J. F., Swindale N. V. (2005) Polytrodes: high density silicon electrode arrays for large scale multiunit recording. J. Neurophys. 93 (5): 2987-3000. doi: 10.1152/jn.01023.2004 PMID: 15548620

Lefebvre, J.L., Zhang, Y., Meister, M., Wang, X., and Sanes, J.R. (2008) Gamma-Protocadherins regulate neuronal survival but are dispensable for circuit formation in retina. Development 135:4141-4151. doi: 10.1242/dev.027912 PMID: 19029044

Li N., Chen T. W., Guo Z. V., Gerfen C. R., Svoboda K., (2015). A motor cortex circuit for motor planning and movement. *Nature* 519(7541):51-56. doi: 10.1038/nature14178 PMID: 25731172

Mizuseki K, Diba K, Pastalkova E, Teeters J, Sirota A, Buzsáki G. (2014) Neurosharing: large-scale data sets (spike, LFP) recorded from the hippocampal-entorhinal system in behaving rats. F1000Res. 3:98. doi: 10.12688/f1000research.3895.2 PMID: 25075302

Mizuseki K, Buzsáki G. (2013) Preconfigured, skewed distribution of firing rates in the hippocampus and entorhinal cortex. Cell Rep. 4(5):1010-21. doi: 10.1016/j.celrep.2013.07.039 PMID: 23994479

Mizuseki K, Diba K, Pastalkova E, Buzsáki G. (2011) Hippocampal CA1 pyramidal cells form functionally distinct sublayers. Nature Neuroscience 14(9):1174-81. doi: 10.1038/nn.2894 PMID: 21822270

Mizuseki K, Sirota A, Pastalkova E, Buzsáki G. (2009) Theta oscillations provide temporal windows for local circuit computation in the entorhinal-hippocampal loop. Neuron 64(2):267-80. doi: 10.1016/j.neuron.2009.08.037 PMID: 19874793

Pastalkova E, Itskov V, Amarasingham A, Buzsáki G. (2008) Internally generated cell assembly sequences in the rat hippocampus. Science. Sep 5;321(5894):1322-7. doi: 10.1126/science.1159775 PMID: 18772431

Peron, S., Freeman, J., Iyer, V., Guo C., Svoboda, K. (2015) A Cellular Resolution Map of Barrel Cortex Activity during Tactile Behavior. Neuron May 6; Vol 86, Issue 3, 783–799. doi: 10.1016/j.neuron.2015.03.027 PMID: 25913859

Zhang, Y.F., Asari, H., Meister, M. (2014); Multi-electrode recordings from retinal ganglion cells. CRCNS.org. http://dx.doi.org/10.6080/K0RF5RZT