The Neuron Detection and Signal Extraction Platform: Introduction to NDSEP Version 1.0[*]

Yaesop Lee, Jing Xie, Eung-Joo Lee,
Rong Chen, and Shuvra S. Bhattacharyya

Document Version: April 10, 2020

This document provides instructions on setting up NDSEP, the *Neuron Detection and Signal Extraction Platform*. For additional information related to the NDSEP package, we refer the reader to the website of the [B]altimore / [C]ollege Park [N]euro[M]odulation (BCNM) Project [1].

1 Installation

This section provides information on installing ("setting up") NDSEP. The following steps outline the installation process. NDSEP depends on a number of software packages, called DICE, LIDE, and Welter, which are part of the DSPCAD Framework [2]. These three packages are bundled within NDSEP, and are installed automatically as part of the process of setting up NDSEP.

Operating system: NDSEP compiles on the macOS and Linux platforms. It has been tested on MacOS Catalina (version 10.15.4), and Ubuntu Linux (version 18.04.1).

OpenCV dependency: Before setting up NDSEP, OpenCV-3 must be installed. Please ensure that OpenCV-3 is installed before proceeding with the setup instructions in the remainder of this section. It is recommended that a version of OpenCV-3 is installed that is *no later than Version 3.4*. On MacOS, NDSEP has been tested with OpenCV versions 3.1, 3.3.1, and 3.4, and on Ubuntu, NDSEP has been tested with OpenCV version 3.3.1.

The setup process for NDSEP consists of the following steps:

1. Download the NDSEP Package from
   `https://dspcad-www.iacs.umd.edu/bcnm/packages/ndsep.tar.gz`

   Then unpack the archived download file `ndsep.tar.gz` (this will result in a single directory called `ndsep`). Place this `ndsep` directory in the directory location where you want it to reside. This location is referred to in the remainder of this document as your *NDSEP installation directory*. For example, if one has placed the downloaded, unarchived `ndsep` directory in `/users/me/import/applications`, then the NDSEP installation directory is: `/users/me/import/applications/ndsep`.

2. Open a `Bash` shell, and `cd` (change your current working directory (CWD)) to your NDSEP installation directory.

3. Run the following command at the `Bash` prompt to install the software:

   ```
   source setup/setup
   ```

   The command above will by default create a `dspcad_user` directory within your home directory if such a directory does not already exist. This directory is created by the DSPCAD Framework software mentioned above, and is used also by NDSEP. In particular, NDSEP uses a subdirectory of `dspcad_user`

[*]Technical Report DSPCAD-TR-2020-01, Maryland DSPCAD Research Group, Department of Electrical and Computer Engineering and Institute for Advanced Computer Studies, University of Maryland at College Park. April 10, 2020. Y. Lee, J. Xie, E.-J. Lee, and S. S. Bhattacharyya are with the Department of Electrical and Computer Engineering and Institute for Advanced Computer Studies, University of Maryland at College Park, USA, Email: {`yaesop, jingxie, elee1021, ssb`}`@umd.edu`. R. Chen is with the Department of Diagnostic Radiology and Nuclear Medicine, University of Maryland at Baltimore, USA, Email: `rchen@umm.edu`.

called `lide_user` since the NDSEP package builds most directly upon the LIDE (Lightweight Dataflow) package within the DSPCAD Framework.

2 Startup

NDSEP is designed to be used within a `Bash` shell. To use NDSEP in a given `Bash` session, one must first start up NDSEP within that session. Starting up NDSEP involves loading necessary environment settings so that you can use all of the features in NDSEP.

To start up NDSEP, follow these steps:

1. Start a `Bash` shell.
2. CD to your LIDE user directory:

   ```
   cd ~/dspcad_user/lide_user
   ```

3. Run the following command:

   ```
   source startup/ndsep_startup
   ```

As a basic test of the startup process, one can run the `ndsversion` command, which takes no arguments, from the enclosing `Bash` session after NDSEP has been started up. If NDSEP has been properly set up and started up, the `ndsversion` command should execute and produce a brief message on standard output that gives the version number and other basic background information for the corresponding installation of NDSEP.

3 Input

The core functionality of the NDSEP package is encapsulated in a utility called `ndsextract`, which extracts neural signals from a given sequence of calcium-imaging-based image frames. Usage of the `ndsextract` utility is demonstrated in Section 5 and explained in Section 6. In this section and the following section, we describe the input and output to `ndsextract`, respectively.

The input to `ndsextract` is a sequence of image frames with each image frame stored in a separate file. The supported file types for the input frames are PNG and TIFF. The input frames in a given image sequence should have the same file type, and should have the file name suffix `png` or `tif` depending on the file type. The file name suffix `tiff` is also acceptable for TIFF format files. The input frames in the given sequence should be placed in the same directory, and named with the same file name prefix, which we refer to symbolically as `<inprefix>`. Similarly, we refer symbolically to the relevant file name suffix for the input image frames as `<insuffix>`.

Each input image frame should be stored in a distinct file whose file name is of the form `<inprefix>_<index>.<insuffix>`, where `<index>` is a positive integer that represents the position of the frame in the temporal ordering of the input images. Lower index values represent earlier frames in the temporal ordering. The index values can start at any positive integer N, and the complete set of indices must form a contiguous set of integers $N, (N+1), \ldots, (N+I-1)$, where I is the total number of images in the input image sequence.

For example, suppose that we a have dataset consisting of $I = 200$ image frames where `<inprefix>` is `dataset5`, `<insuffix>` is `<png>`, and the starting image frame index is $N = 5$. Then the images should be stored within the same directory in a set of 200 files named `dataset5_5.png`, `dataset5_6.png`, ..., `dataset5_204.png`.

4 Output

The output of `ndsextract` consists of two parts — a summary of the extracted neural signals, and a sequence $F_1, F_2, \ldots, F_I$ of image frames that show the positions of the detected neurons. Recall that I denotes the total number of input image frames.

Each output image frame is stored by `ndsextract` in a distinct file whose file name is of the form `<index>_out.<outsuffix>`, where `<index>` is again a positive integer. The filename extension `<outsuffix>` is determined as part of the argument list to `ndsextract` (see Section 6) and can be either `png`, `tif` or `tiff` as with `<insuffix>`. The output filename extension `<outsuffix>` (and the assumed file format) need not be the same as `<insuffix>`.

Each output frame F_i displays the positions of the detected neurons in the ith earliest input image frame. The file name associated with F_i is `<index(i)_out>.<outsuffix>`, where `<index(i)>` is the file name index associated with the ith earliest input frame. All of these image output files are stored by `ndsextract` in the directory `ndsoutput`, which is created automatically by `ndsextract`. `ndsextract` aborts with an error message if `ndsoutput` already exists in the CWD.

Note that the set of detected neurons shown in each F_i is cumulative — that is, it includes all of the neurons that have been detected in earlier frames. Once a neuron is detected by `ndsextract` in a given frame, it is tracked in all subsequent frames of the given image sequence.

Fig. 1 shows an example of an output image file produced by `ndsextract`. In this example, each green circle indicates the position of a detected neuron. A total of 14 detected neurons can be seen in this example.

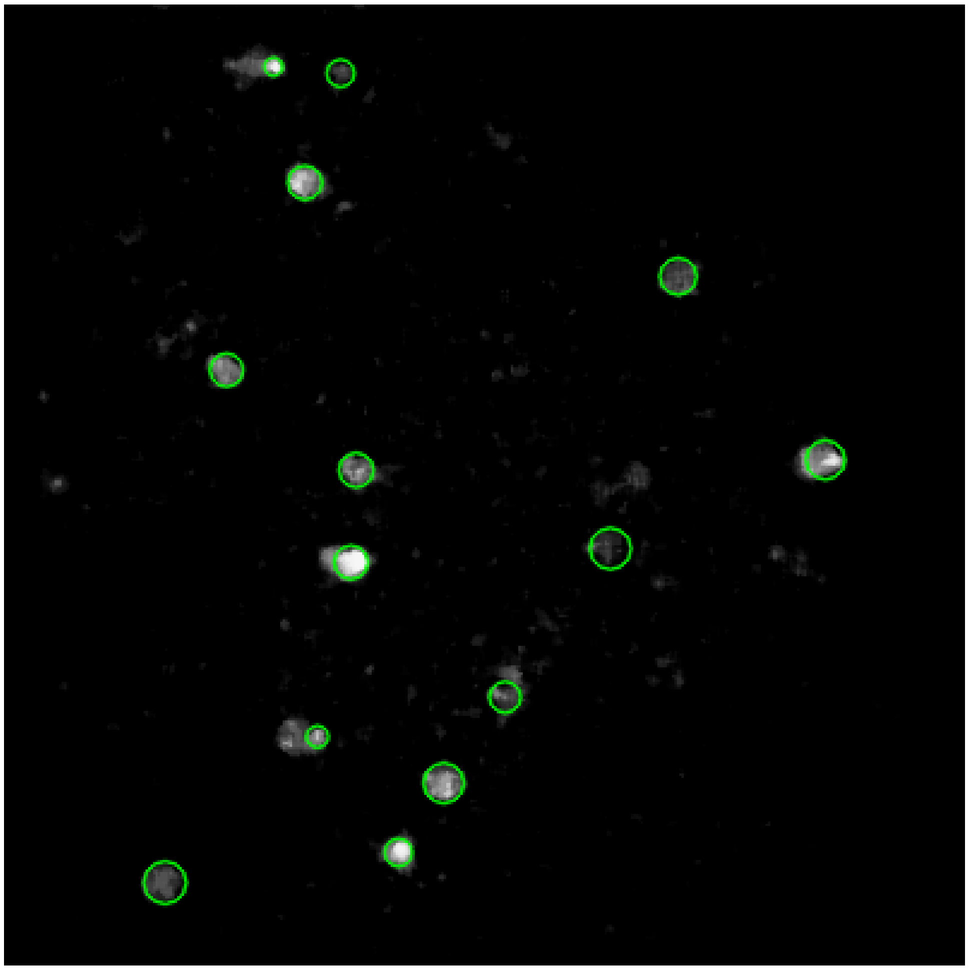

Figure 1: An example of an output image file produced by `ndsextract`.

In addition to producing the output image frames described above, `ndsextract` also produces a file called `signals.txt`, which stores a representation of the signal extracted from each neuron. The `signals.txt` file is placed in the CWD — that is, the parent directory of the `ndsoutput` directory. The `signals.txt` file contains a set of floating point numbers that correspond to values of individual elements within a matrix R having dimensions $I \times X$, where X is the total number of extracted neurons across all input image frames.

The ith line in `signals.txt` corresponds to the ith row of R, and for $j = 1, 2, \ldots, X$, the jth value listed on line i corresponds to matrix element $R[i][j]$. Each matrix value $R[i][j]$ gives the signal intensity value of the jth neuron as measured from the ith input image frame.

5 A Demo

To retrieve a sample dataset that is included as part of `ndsextract`, run the following command:

```
ndsget sampledata
```

The `ndsget` command is provided to copy into the CWD selected examples, such as datasets and demos, that are included within `ndsextract`. Presently, only one example identifier (`sampledata`) is recognized by `ndsextract`. We anticipate that more examples will be added in future release versions, which can then be retrieved in a similar way using `ndsget`.

The execution of the `ndsget` command shown above results in a sample dataset being placed within a directory called `demodataset` in the CWD. Neuron detection and signal extraction can then be performed on this dataset using `ndsextract` using the following command:

```
ndsextract ./demodataset frame 1 png tif 1800
```

The output resulting from this command will consist of 1801 files (1800 `tif` files along with the `signals.txt` file). The `tif` files will be placed in a subdirectory within the CWD called `ndsoutput`, while the `signals.txt` file will be placed in the CWD itself. See Section 4 for details on the contents of the generated `ndsoutput` directory. Details on the usage of `ndsextract` are presented in Section 6.

6 `ndsextract` Usage

The `ndsextract` command takes six arguments. The general form for using `ndsextract` is as follows.

```
ndsextract <indir> <inprefix> <startindex> <insuffix> <outsuffix> <framecount>
```

The meanings of the arguments are as follows.

- `<indir>`: The directory in which the image files for the input frames are stored.
- `<inprefix>`: The common prefix for the input frame file names (see Section 3).
- `<startindex>`: The starting index for the frame indices associated with input and output image file names. This corresponds to N, which was defined in Section 3.
- `<insuffix>`: The file name suffix for the input frames. Admissible values are `png`, `tif`, and `tiff`. This argument also specifies the input image format.
- `<outsuffix>`: The file name suffix for the output frames. Admissible values are again `png`, `tif`, and `tiff`. This argument also specifies the format in which `ndsextract` will store its output images.
- `<framecount>`: The number of frames in the input image sequence. This corresponds to I in earlier sections of this document.

For a concrete example of `ndsextract` usage, see Section 5.

Acknowledgments

This research was supported in part by the National Institutes of Health (NIH) / National Institute of Neurological Disorders and Stroke (NINDS) under Grant No. R01NS110421 and the Brain Research through Advancing Innovative Neurotechnologies (BRAIN) Initiative.

References

[1] “BCND project website,” 2020, https://dspcad-www.iacs.umd.edu/bcnm/index.html.

[2] S. Lin, Y. Liu, K. Lee, L. Li, W. Plishker, and S. S. Bhattacharyya, “The DSPCAD framework for modeling and synthesis of signal processing systems,” in *Handbook of Hardware/Software Codesign*, S. Ha and J. Teich, Eds. Springer, 2017, pp. 1–35.