NeuralSorter Basics

July 7, 2020

Note: This document covers the basic information to use NeuralSorter Beta version

1 Dependencies

Numpy, Matplotlib, Tensorflow==2.1.0, PyQT5, sklearn

If you are using Anaconda and it doesn't support support Tensorflow 2.1.0, create a new environment and run:

conda install -c anaconda -keras-gpu

2 Project aims

NeuralSorter is a project for semi-automatic spike cleaning and sorting. It includes a cleaning and sorting pipeline together with a GUI for data curation. In addition, it has been developed using a modular approach that makes it easy for the user to substitute any step of the data processing (data type to load, cleaning network, sorting algorithm) for any custom designed approach, while continuing to use the GUI. In addition, complementary scripts to facilitate spike sorting can be easily added to PySorter, together with the ones already included.

3 Overview

PySorter is a semi-automatic spike cleaner and sorter. It works at three levels: First, it normalizes the data and discard abnormal shapes to remove noise events using a CNN; this network can be trained with previously cleaned data to adapt it to the specific recording of each setup. Then, putative units are sorted using a combination of an autoencoder to reduce dimensionality and clustering using Gaussian Mixture Models. This second step can be repeated to subsequently split desired units. At last, all units are visualized for manual curation; in addition, different functions are provided to facilitate the data curation (acor, data visualization, ISI).

```
spike_dict['ExperimentID']          #ID of each file to sort multiple
                                    #files simultaneously
spike_dict['ChannelID']             #Channel to which the event belongs
spike_dict['UnitID'].append( 1 )    #Unit to which the event belongs
                                    #(initial = 1)
spike_dict['OldID'].append( None ) #Buffer that allows the undo function
                                    #(initial = None)
spike_dict['TimeStamps']            #Timestamp of the event
spike_dict['Waveforms']             #Waveform of the event
```

Table 1: Data format

3.1 Data formatting

In order to use NeuralSorter, data has to be transformed from its original format to the dictionary spike_dict, which carries the necessary information to curate multiple files simultaneously.

The dictionary spike_dict includes 6 lists. Each events adds a value (or values in the case of waveforms) in the same position to each of the them, as indicated in the figure. As an example, the first event will have the information about the experiment in spike_dict['ExperimentID'][0], its waveform will be stored in spike_dict['Waveforms'][0], and the same with the other features. The next event will fill the position [1] of each list in spike_dict, etc. Note that 'UnitID' and 'OldID' are always filled with "1" and "None" respectively. In order to use the pretrained network with the original structure, the waveforms have to consist on a vector ranging from 1ms before to 2ms after the threshold transition of the spike, with 20 KHz sampling rate (60 points).

4 NeuralSorter interface

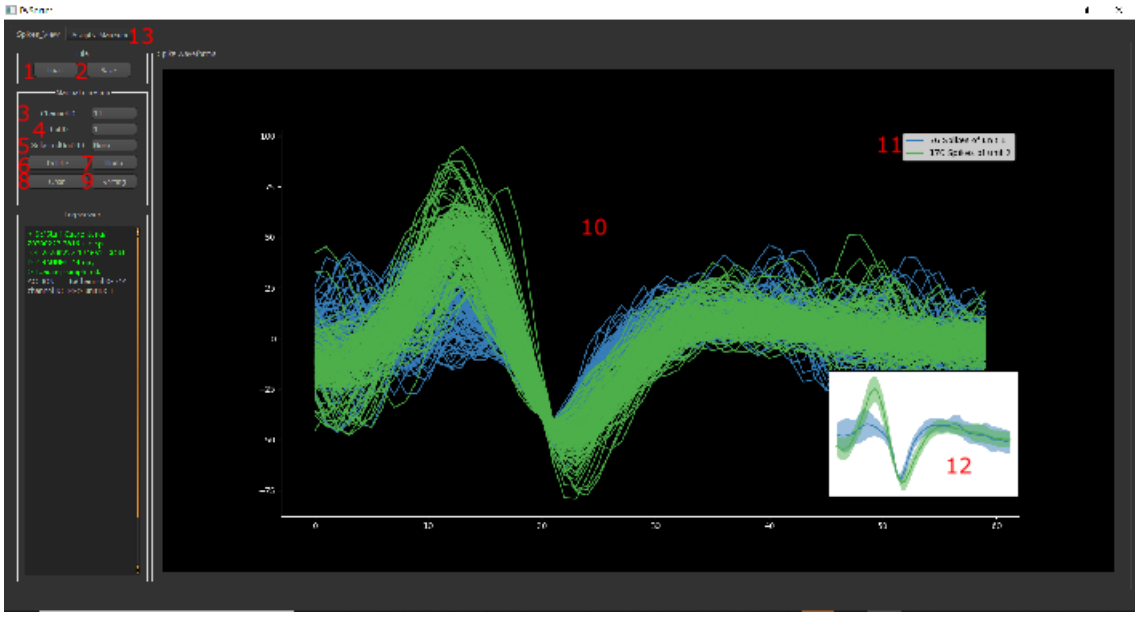

Figure 1: PySorter interface

NeuralSorter uses an interactive interface to perform the spike sorting. In 1 the main elements of the interface are shown.

1) Loading button.

2) Saving button.

3) Index with all the channels in the recording.

g 4) Index with all the units in the present channel.

5) Indicates to which channel to send the selected events.

6) Sends the selected events to the unit selected in 5 (equivalent to Alt+[number of units]) // Noise units have the Unit nº = 0 assigned.

7) Undoes the last action (Does not stack).

8) Loads the CNN and uses it to clean all electrodes. The events considered as noise will be sent to the "noise" unit of each channel.

9) Applies the sorting pipeline to the data displayed in the axis.

10) Main screen.

11) Colour code of each units and number of events.

12) Average wave shape of each unit.

13) Utilities.

4.1 Loading data

You can load multiple files to sort simultaneously, with Experiment ID keeping track of the origin file for each event. If you want to load multiple files, the order on which you click them will be the order in the 'ExperimentID' list. If the load button is used again, it will erase the previous spike_dict, so be sure to save your data before loading new files.

5 Shortcut keys

```
Ctrl+Up/Down ->
#move across channels.
Shift+Up/Down ->
#move across units on each channel.
Alt+[0,1,2,3,4,5,6,7,8,9] ->
#sends selected spikes to the specified
# unit, 0 unit is used for noise.
Ctrl+c ->
#clean spikes from noise for all the loaded waveforms.
Ctrl+s ->
#Sorting of current visualized waveforms.
Ctrl+d ->
#Send selected waveforms to noise, equivalent to Alt+0.
Ctrl+z ->
#Undo, only available for the last modification
```

Table 2: Shortcut keys of NeuralSorter interface

6 Algorithms

6.1 Cleaning

Cleaning of non-spike events is performed with a CNN trained for such porpoise. This enables a faster spike sorting in those cases on which the signal-to-noise ratio is not good or small multiunits should be recovered. An additional script "train_cleaner_script" is provided to train your own CNN if you have already sorted data that can be representative of your recording setup. Events should be normalized beween [0 1] before using them for the training, as the CNN is designed to distinguish events based on their shape at this stage of the pipeline.

6.1.1 Modifying the cleaner

By default, a pretrained model is loaded in /CLEANER/cleaner.py. The model can be easily substituted there by any other type of ANN that a user of Neu-

ralSorter may train. In order to do so, the name of the trained network has to be provided in the corresponding section of the script NeuralSorter/GUI/-GLOBAL_CONSTANTS.py. Loss function, optimizer and bactch size can also be changed there.

```
''' THE DEEP LEARNING MODEL WHICH MAKES THE INFERENCE
 BETWEEN NOISE AND SPIKE EVENTS '''

CLEANER_DEEPL_H5_MODEL = "./CLEANER/model_to_load.h5"
LOSS='categorical_crossentropy'
OPTIMIZER='adam'
BATCH_SIZE = 16
```

Table 3: How to change the network used to clean the data in GLOBAL_CONSTANTS

An additional script to generate new CNNs is included in NeuralSorter, named "train_cleaner_cnn". This script serves to train new CNNs using previously curated data from the user's lab, thus customizing a personalized network for the cleaning of future data. Note that to retrain a network with the original structure, it asks for a trace ranging from 1ms before to 2ms after the threshold transition of the spike, with 20 KHz sampling rate (60 points). Alternatively, a similar class can be created and called from the main script "SPIKE_SORTING". In that case, rescrictions in the sampling rate or waveshape can be determined by the NeursalSorter user.

6.2 Sorting

In order to sort the units, their dimensionality is reduced using an autencoder and then they are clusterized using a GMM. The number of clusters is based stablished using information-theoretic criteria (BIC). Clustering is performed on the events ploted in the axis (10). Recursive clustering and/or re-clustering can be performed if considered benefitial.

6.2.1 Modifying the sorter

By default the previously described pipeline is used, stored in ./SORTER/-sorter.py. If a user decides to change it by any other desired sorting pipeline while still using NeuralSorter-GUI, it can be implemented in sorter.py. Alternatively, a similar class can be created and called from the main script "SPIKE_SORTING".

6.3 Utilities

Several functions are available to help in the sorting procedure in the "Scripts_Manager" tab. They operate using the events that are plotted in the main axis (10) when

the script is executed. Available scripts are displayed on the left side. When one script is selected, it can be read and modified in the "Python raw code" window.

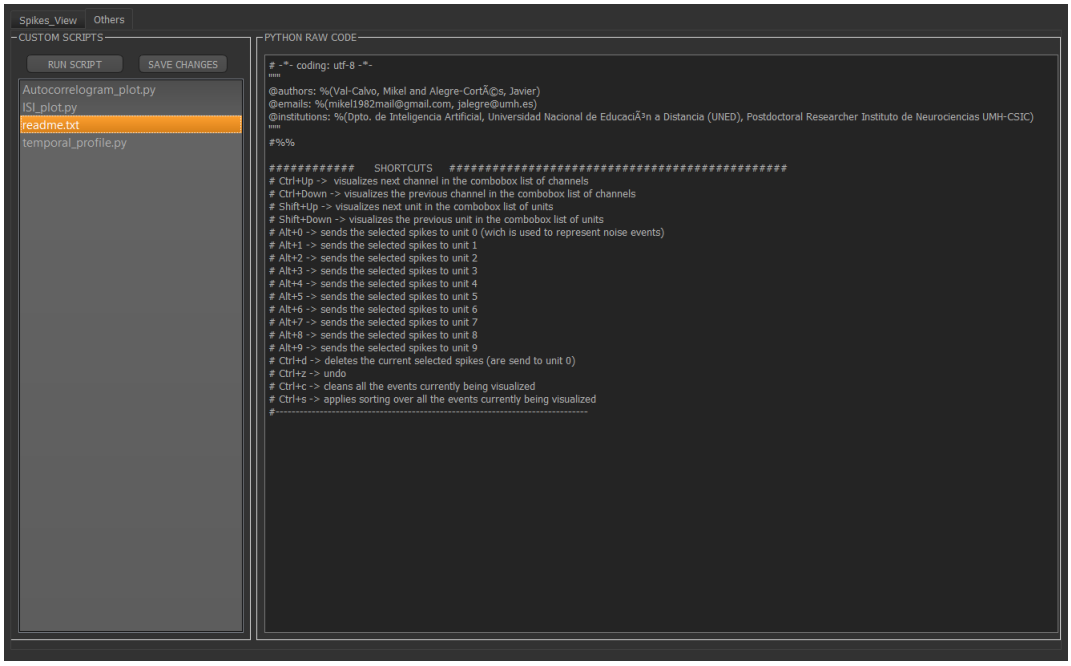

Figure 2: Utilities interface

At this moment, ISI, autocorrelogram and temporal representation of the spikes are included. Source code can be edited in the "Python raw code" window. Additional scripts with new analysis/visualization tools can be easily added including them in the folder "AUXILIAR_CODE".