PyNeuroTrace - Python code for Neural Timeseries

14 April 2024

Summary

Modern techniques in optophysiology have allowed neuroscientists unprecedented access to neuronal activity *in vivo*. The time series datasets generated from these experiments are becoming increasing larger as new technologies allow for faster acquistion rates of raw data. These fluorscent recordings are being made with an ever expanding library of indicators encoding calcium, voltage, neurotransimitter and neuromodulator activity. These signals generated from these fluorscent bioindicators contain the information of the underlying neuronal activity but all have unique molecular kinetics and inherit signal-noise ratios which must be taken into account during singal processing. The development of pyNeuroTrace, an open-source Python library, was made to aid in the processing of these neurnal signals which must be filtered with these unique aspects in mind before analysis can be completed.

Statement of need

Many neuroscience labs that use optophysiological methods, such as two-photon microscopy or fibrephotomotery, frequently must rewrite and maintain common functions and filters needed to analysis the raw recordings. Furthermore, many technique and algorithms for signal processing are scattered throughout the literature and are frequently implemented programing languages other than Python. `pyNeuroTrace` meets the need of a time series analysis written purely in Python for neuronal activity. Our package is a collection of filters and algorithms implemented in a genralizable manner for time series data in either 1D-arrays or a collection of recording in a 2D-arrays. Additionally, with the increase in aquistion rate of new imaging techniques, we have implementations of these algorithms using GPU compatiable code to increase the speed in which the techniques can be applied to larger datasets collected at kilohertz rates.

Signal Processing

DeltaF/F

There are several methods for calculating the change of intensity of a fluorscent trace (Grienberger et al. 2022). We implemented the method described by Jia *et al* for the calculation of $\Delta F/F$ normalizes the signal to a baseline which helps with bleaching or other changes that occur over time which influence the detection or magnitude of events in the raw signal(Jia et al. 2010). This implementation includes several smoothing steps to help with shot noise(Jia et al. 2010). In short, F_θ is calculated by finding a the minmum signal in a window of the rolling average of the raw signal. Then ΔF is calculated by the difference in the raw signal and F_θ, which is then divided by F_θ to get the trace for $\Delta F/F_\theta$. This $\Delta F/F_\theta$ signal is optionally smoothed using an exponetially wieghted moving average (ewma) to further remove shot noise. Jia *et al* defined rolling average with the following equation:

$$\bar{F} = \left(\frac{1}{\tau_1}\right) \int_{x-\tau_1/2}^{x+\tau_1/2} F(\tau)\, d\tau$$

The F_θ is defined as uing a second time constant, τ_2, to define a window for the search for the minimum value to be used as a rollowing baseline:

$$F_\theta(t) = min(\bar{F}(x))|t - \tau_2 < x < t$$

Thus $\Delta F/F$ is:

$$\Delta F/F = \frac{F(t) - F_\theta}{F_\theta}$$

The two time constants, τ_1 and τ_2, can be selected by users. Modifying the these parameters will have a dramatic influence on the output signal.

Okada Filter

We implement the Okada Filter in Python(Okada, Ishikawa, and Ikegaya 2016). This filter is designed to filter shot-noise from traces in low-signal to noise paradigms, which is common for calcium imaging with two-photon imaging where the collected photon count is low and noise from PMT can be nontrivial.

Nonnegative Deconvolution

`pyNeuroTrace` also has an implementation of nonnegative deconvolution (NND) to be applied to photocurrents to reduce noise in raw time series recordings (Podgorski and Haas 2012). These alogrithm can also be used to aid in the detection of events associated with neuronal activity which follow similiar decays as photocurrents from detects, as small events in fluorscent imaging are often obfuscated by noise in the signal(Podgorski and Haas 2012).

Event Detection

The event detection module uses several strategies to indentify neuronal activity in time series datasets. These methodologies have been previously discussed and compared by Sakaki *et al* (Sakaki et al. 2018). These include to generalizable methods and one that requires prior knowledge of recorded event shape. The generalizable methods include filtering the signal through an exponentially weighted moving average (ewma) or cumulative sum of movement above the mean (cusum). The final filter is a match filter which finds the probablity of the trace matching a prior defined shape, such as one described by an exponetional rise and decay of calcium signal.

Matched Filter

Visualization

`pyNeuroTrace` has several in built visualization tools. 2D arrays of neuronal timeseries can be displayed as heat maps Figure 1 or as individual traces Figure 2. The heatmap is a useful visualization tool for a larger number of traces, additionally at the bottom of the plot the stimuli timing is displayed if provided Figure 1. This allows for quick visual inspection of activity from a population of neurons or from

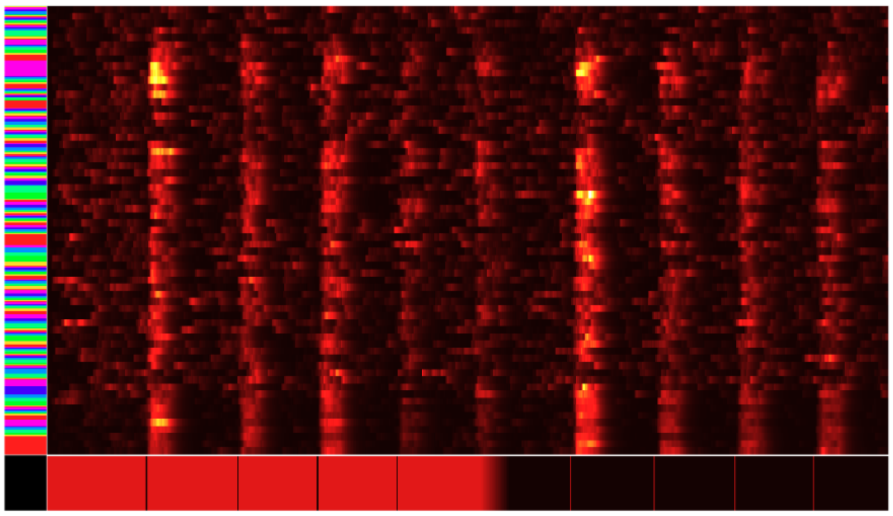

Figure 1: Caption for example figure.

and referenced from text using Figure 1 Figures can be included like this:

and referenced from text using **??**.

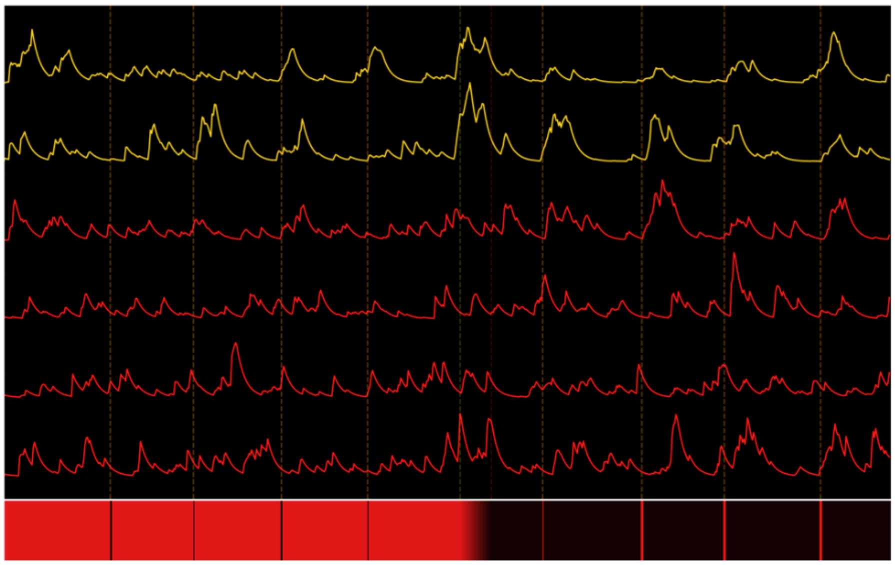

Figure 2: Caption for example figure.

One of these in-built visualizations is specefic to the data structure generated by a custom acusto-optic

GPU Acceleration

Several of the filters in `pyNeuroTrace` have been rewritten to be almost entirerly vectorized in their calculations. The benefit being a noticable difference in the performance for larger time series. These vectorized versions gain further speed by being excuted on a GPU using the Cupy Python library (Okuta et al. 2017). To excuted these versions the filters can be imported from the module, `pyneruopyneurotrace.gpu.filters`, and a CUDA compatiable graphics card is needed. This functionality is becoming increasingly important as acquisition rates increase for kilohertz imaging of activity can generate arrays hundreds of thousands of datapoints in length in just a few minutes. Figure 4 shows the difference in calculating arrays of various sizes using either the CPU or vectorized GPU based approach of the dF/F function. The CPU used in these calcultions was a Intel i5-9600K with six 4.600GHz cores, the GPU was a NVIDIA GeForce RTX 4070 with CUDA Version 12.3.

To vectorize the functions several where modified. For example the EMWA used to smooth the dF/F signal as described by Jia *et al* was changed to an approximation using convolution with an exponetional function. The kernel used to perform is defined as:

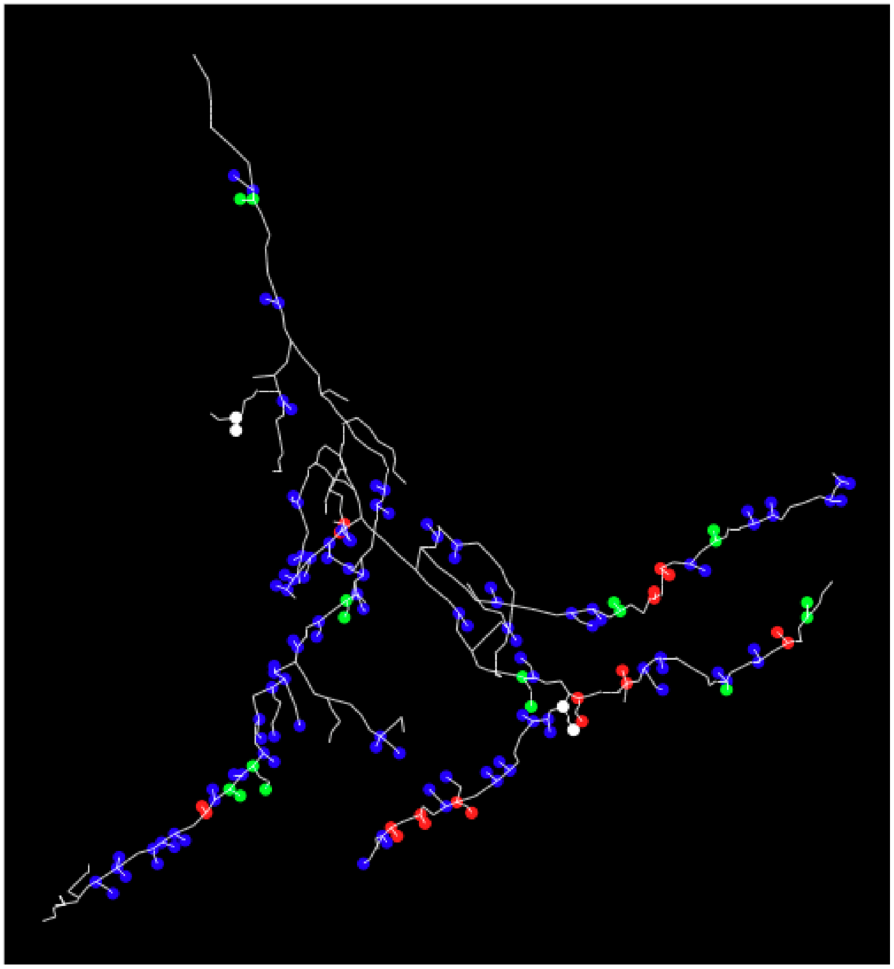

Figure 3: Example of lab specific visualation of

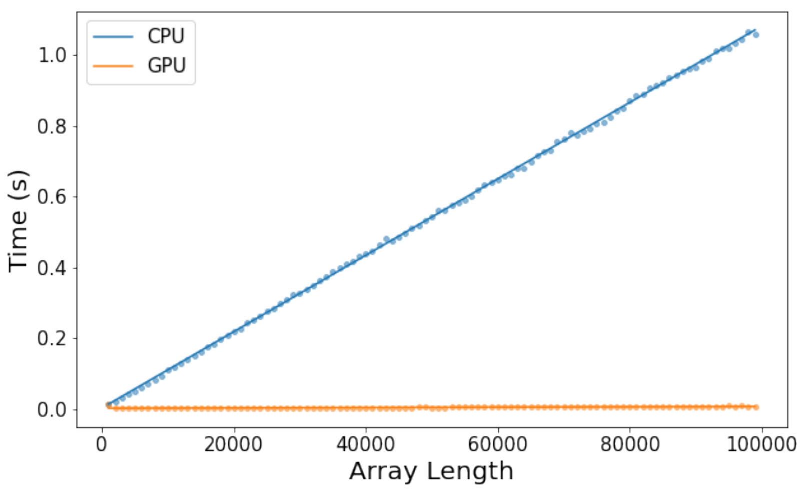

Figure 4: Comparison between dF/F with EWMA calculations for different array sizes.

$$w[i] = \begin{cases} \alpha \cdot (1-\alpha)^i & \text{for } i = 0, 1, 2, \ldots, N-1 \\ 0 & \text{otherwise} \end{cases}$$

Where α is defined as:

$$\alpha = 1 - e^{-\frac{1}{\tau}}$$

τ where is a user selected time constant which is translated into number of samples. N is a window parameter for the kernal calcuated using α:

$$N = \left\lfloor -\frac{\log(10^{-10})}{\alpha} \right\rfloor$$

This allows a filter for smaller values that have a minisclue influence on the weighted average. The kernel needs to be normalized to produce smoothing with the same output value as the non-vectorized impementation:

$$w[i] \leftarrow \frac{w[i]}{\sum_{j=0}^{N-1} w[j]}$$

The normalized kernel is then convolved with the dF/F signal, d:

$$c[k] = \sum_{i=0}^{N-1} w[i] \cdot d[k-i]$$

This convolved signal, c is then normalized to the cummulative sum of the exponetial kernel:

$$n[j] = \sum_{i=0}^{j} w[i]$$

$$emwa = c[i]/n[i]$$

Differences between the CPU and GPU implenetations of the EWMA for an array of ranom values have been plotted Figure 5. These were generated from the same array using the respective decays for either implementation using the time constant of 50 miliseconds and a sampling rate of 2kHz. The difference between the two outputs typically range in magnitude from 1e-16 to 1e-12 depending on user parameters. These discrepancies can also be attributed to differences in floating point number accuracy between CPU and GPU calcultations.

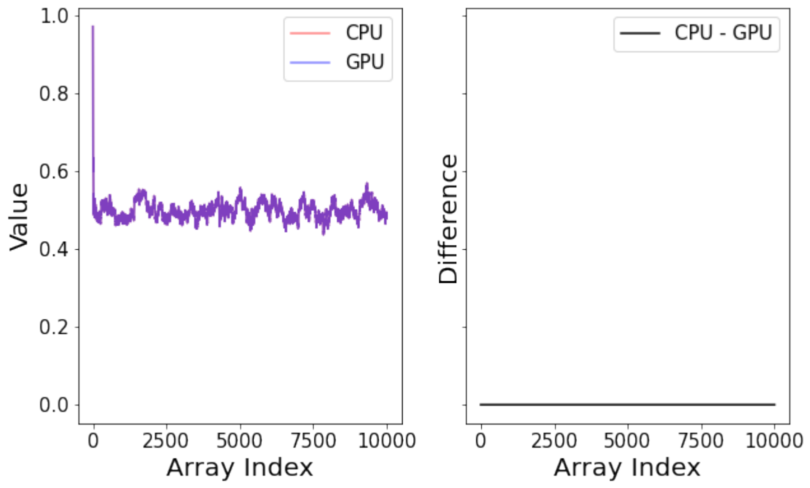

Figure 5: Overlay of the EWMA calculations using the CPU implementation and GPU approximation in red and blue. The difference in values from the output is also plotted.

Acknowledgements

The development of this software was supported by funds from the Canadian Institutes of Health Research (CIHR) Foundation Award (FDN-148468).

References

Grienberger, Christine, Andrea Giovannucci, William Zeiger, and Carlos Portera-Cailliau. 2022. “Two-Photon Calcium Imaging of Neuronal Activity.” *Nature Reviews Methods Primers* 2 (1). https://doi.org/10.1038/s43586-022-00147-1.

Jia, Hongbo, Nathalie L Rochefort, Xiaowei Chen, and Arthur Konnerth. 2010. “In Vivo Two-Photon Imaging of Sensory-Evoked Dendritic Calcium Signals in Cortical Neurons.” *Nature Protocols* 6 (1): 28–35. https://doi.org/10.1038/nprot.2010.169.

Okada, Mami, Tomoe Ishikawa, and Yuji Ikegaya. 2016. “A Computationally Efficient Filter for Reducing Shot Noise in Low S/N Data.” *PLOS ONE* 11 (June): e0157595. https://doi.org/10.1371/journal.pone.0157595.

Okuta, Ryosuke, Yuya Unno, Daisuke Nishino, Shohei Hido, and Crissman Loomis. 2017. “CuPy: A Numpy-Compatible Library for Nvidia Gpu Calculations.” In *Proceedings of Workshop on Machine Learning Systems (Learningsys) in the Thirty-First Annual Conference on Neural Information Processing Systems (Nips)*. http://learningsys.org/nips17/assets/papers/paper_16.pdf.

Podgorski, Kaspar, and Kurt Haas. 2012. “Fast Non-negative Temporal Deconvolution for Laser Scanning Microscopy.” *Journal of Biophotonics* 6 (2): 153–62. https://doi.org/10.1002/jbio.201100133.

Sakaki, Kelly D. R., Patrick Coleman, Tristan Dellazizzo Toth, Claire Guerrier, and Kurt Haas. 2018. “Automating Event-Detection of Brain Neuron Synaptic Activity and Action Potential Firing in Vivo Using a Random-Access Multiphoton Laser Scanning Microscope for Real-Time Analysis.” In *2018 40th Annual International Conference of the Ieee Engineering in Medicine and Biology Society (Embc)*. IEEE. https://doi.org/10.1109/embc.2018.8512983.