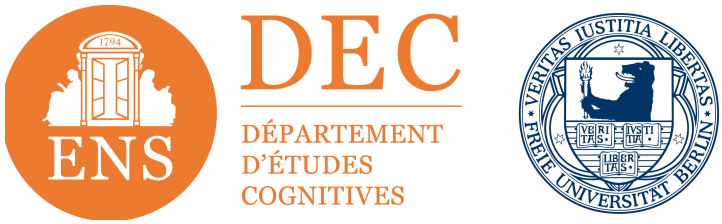
CIDA - a Calcium Imaging Deconvolution API
Master Thesis

Maximilian Ernst Otto Pfeiffer

December 24, 2017



Contents

1 Abstract

Numerous assays geared towards capturing neural activity exist. They range from patch-clamp methods, targeting singular cell membrane potentials, to larger scale techniques capturing the activity of neural populations and even whole brain regions, such as local field potential (LFP) recordings and electroencephalography (EEG), respectively.
Unfortunately, life science in general and neural research specifically is subject to a strong invasiveness-resolution trade-off: increasingly invasive methods (such as electrophysiological recordings, which require introducing electrodes into the brain) yield increasingly high-resolution data, while less invasive techniques suffer from a loss of resolution[1].
Furthermore, highly invasive methods are more likely to disrupt the system being studied, potentially producing atypical behavior. This is particularly problematic for studies aiming to uncover how neurons process information at the neural circuit level; neuronal populations build very deliberate and precise architectures and exhibit a high level of organization and regulation, necessitating high resolution assays[2].

One promising method is Calcium neuro-imaging(CNI), a widely used technique with many applications in neuroscience. CNI is less invasive than standard electrophysiological recordings, and is able to capture the activity of a relatively high number of neurons simultaneously[3]. This method combines genetically encoded calcium indicators with laser-illuminated two-photon microscopy, and allows for measurement of neural activity of up to thousands of neurons simultaneously[1][4][5].
Determining an underlying action potential train from the observed calcium concentration trace is a non-trivial task because the relationship between the two is complex. There are several publications that tackle this problem and report good results, however they tend to not supply their code or restrict availability to matlab[2][6][7][8][9]. Furthermore, most available implementations employ computationally expensive models, limiting their applicability.
Unlike these approaches, this thesis presents an API using temporal deconvolution to extract spike trains, a method which has been reported to produce good results before[5]. It critically depends on the ability to model the calcium signal response to an action potential, for which I present a novel extraction algorithm, improving the accuracy of deconvolution. This annotation method is based on signal change ratios observed in the data, and can generally be used to label regions of activity in time series.
The code is made available via a public license(GNU GPL), allowing for easy modification and adaptation. Additionally, it contains quantile- and baseline-normalization methods for time series. Also included is a script which can be run from a terminal, requires no installation and can deconvolve input data as-is, without requiring any specific parameters to be adapted.

2 Implementation

2.1 Availability & Requirements

All of the code is published under the GPL(general public license) and is openly available on GitHub[32].
The algorithm was written with python 2.7, the dependencies are the SciPy[33], Pandas[34] and NumPy[35] libraries, which are standard tools for scientific computing in python.
Testing was performed on 32 and 64-bit Unix kernels as well as Windows 7 and 10, and the API performs normally with a python 3.2 interpreter.
The repository contains a runnable script called "main.py" and a second file containing the API's functions, "cal_neuroIm.py".
An additional ipython notebook is in the github repository, and is titled "simroutine.ipynb". It mostly contains code for data simulation, visualizations and extracting data from several file formats, such as .mat.

2.2 Performance

In order to assess the accuracy and reliability of the method, I simulated data by randomizing the free parameters of an alpha kernel, total transient duration and total number of action potentials in the trace.
More specifically, I modeled baseline activity as centered around mean 400 with normally distributed noise of mean 0 and $\sigma = 80$, while AP timings were randomly distributed over the series without any rules preventing overlap of transients. Single AP responses were thus simulated by randomizing the parameters of the alpha kernel (see section **??**):

$$f(t) = A \cdot e^{t/t_A} - e^{t/t_B}$$

Notice that transient length t is another parameter to be randomized. The respective mean and standard deviance of the normal distributions sampled from were: $\{\sigma_t = 120, \mu_t = 400\}, \{\sigma_A = 120, \mu_A = 2000\}, \{\sigma_t A = 30, \mu_t A = 120\}, \{\sigma_t B = 10, \mu_t B = 40\}$.
These ranges were well within those observed in unpublished data by Fani Kokuli (Institut Pasteur, Paris), albeit transient characteristics are generally expected to vary strongly w.r.t to the cell type and specimen at hand[7](see Figure **??**), but retain some consistency grouped as such.
Additionally, a fully annotated dataset including cell-attached spike measurements with simultaneous CNI was made available by the CRCNS data sharing program(Collaborative Research in Computational Neuroscience)[36][37].
The first part of the set consists of two separate imaging runs from the L2/3 primary visual cortex of mice, and captured 11 and 10 cells, respectively.
It contains simultaneous CNI and cell-attached current clamp measurements, and was made available by Dana et al Mar. 2016. Akerboom et al[18] and Chen et al[19] conducted experiments on a diverse range of cells, among which are

in vitro astrocytes and mouse retina, in vivo drosophila adult antennal lobe cells and larval neuromuscular junction, mouse visual cortex as well as zebrafish retina and tectum recordings. In total, their shared data provided an additional 5 sets of recordings with simultaneous patch-clamp measurements.

2.2.1 Temporal resolution

It should be noted that at its core, calcium dynamics lack the speed to be useful for capturing AP timings at an accuracy necessary for precisely timed analyses. This is apparent when regarding the sampling rates typically used for CNI: It reportedly does not make sense to use frame rates above 50Hz, because no significant changes in fluorescence are observed between images[21]
An action potential, however, typically lasts less than 10 milliseconds; if one considers the refractory period to be part of the AP[14]. As a consequence, even in the best case scenario (that is a small, recurring transient surrounded by stable baseline activity, most likely indicating a single burst of action potentials), it is still not possible to determine when exactly between the two frames surrounding the transient onset the action potential was triggered. It is thus an inherent property of CNI that the best temporal resolution possible is determined by the frame rate. Common capturing frequencies like 17Hz take a picture every 58.8ms, which is their smallest distinguishable time difference.
The much more frequent case is accordingly less suitable for precise timing analysis: I am referring to traces with continuous spikes, were it is very likely that multiple APs overlap within a singular transient, meaning that no region of baseline activity separates the transients.
In these cases, general development of overall activity is well captured regardless, and the data can be deconvolved with regards to it. However, determining the exact number of spikes with sufficiently precise timings becomes hard. High-activity data with a corresponding density of APs reduces the accuracy of the deconvolution. Technically, the algebraic approach implemented here provides exact timings, but it is advisable to interpret these as an indicator of general change of activity instead. One such method would be to regard firing rates over time bins, and deriving the change of activity. Two examples of this are presented below, which use heat maps to visualize activity patterns in the captured neurons, see Figure 3 and 5.

Figure 3 displays relative activity of a cell within a trace, and shows deconvolved (below) and patch-clamp-recorded firing rates over 460 frame bins.
It is apparent that the CNI deconvolution approach generally yields less activity than its cell-attached counterpart.
This is especially true for continuous spiking: Long, fairly homogeneous stretches of activity are de-emphasized by the deconvolution method.
Detecting an onset of activity in response to e.g. a stimulus, or conversely a suppression of such, is well possible however.
Figure 5, showing a similar heat map of the data generated by Chen *et al.*,

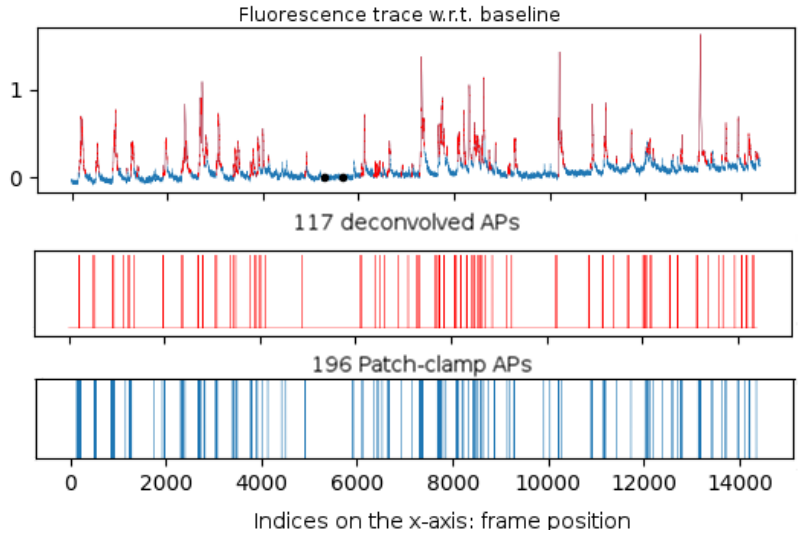


Figure 1: From top to bottom: fluorescence signal recorded in a cell, deconvolved spike train, and spike train as recorded via patch-clamp. Note that while the deconvolution method yielded roughly half the total amount of action potentials as annotated via patch-clamp, the pattern of activity, and change thereof, are almost identical. Data from Chen *et al.*[19].

exhibits comparable properties, as lasting stretches of moderate activity are generally underrepresented.
Regarding the data on a second-scale for visualization, by binning activity in the heat map over 4s windows, conserves patterns of firing rate changes well.

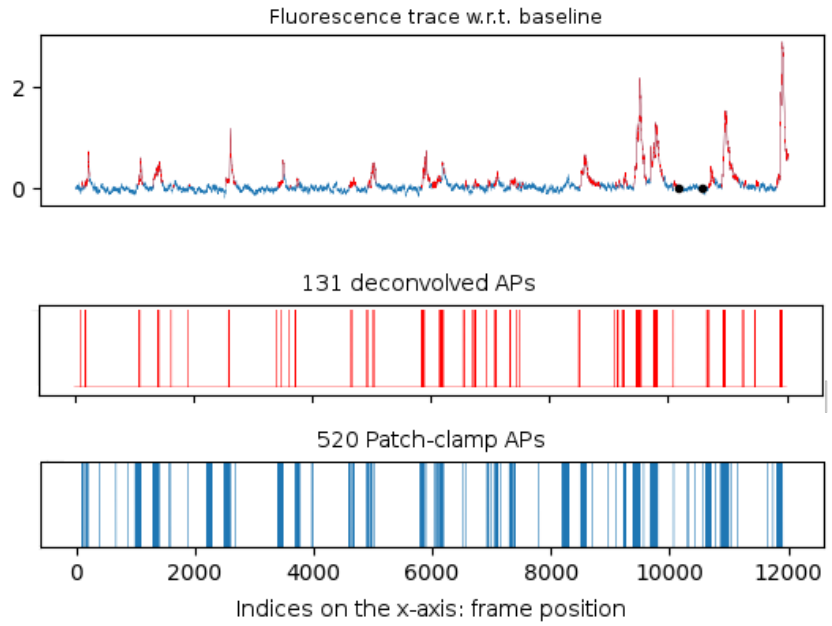


Figure 2: Same depiction as Figure 1, but with data from Akerboom *et al.*[18]. The deconvolution method missed smaller prevalence of activity in between bigger transients, and underestimates the number of spikes contained in these. However, as before, the distribution of activity derived from it very closely resembles those retrieved via patch-clamp.

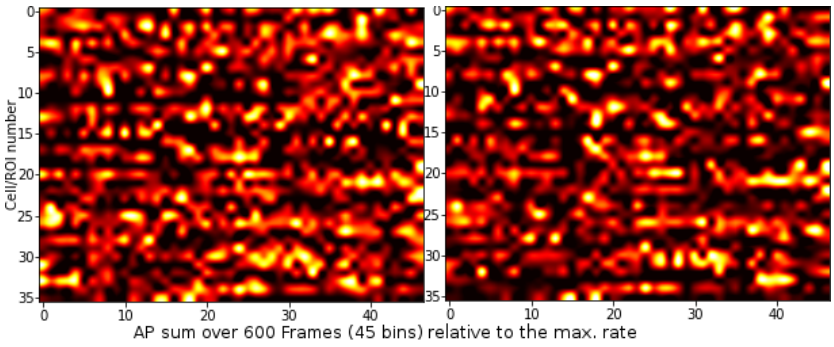


Figure 5: Similar to 3, with each cell/ROI on a horizontal axis. left: electrophysiological measurements, right: deconvolved CNI activity. All firing rates expressed relative to the maximum encountered in each respective cell. Data from Chen *et al.*[19]. Note that the first row corresponds to the data depicted in Figure 1, meaning 14000 frames recorded at 17Hz.

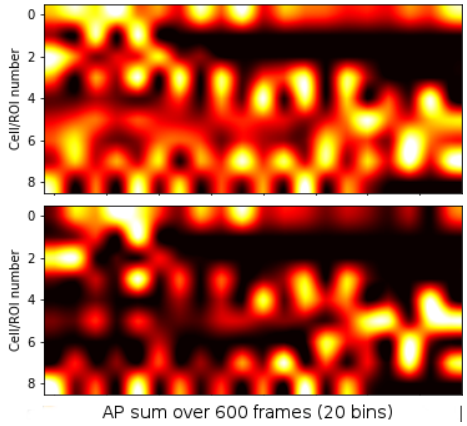


Figure 3: One layer on the horizontal axis corresponds to one ROI in the recordings, while color intensities ('hotter' → higher) encode firing rate relative to the maximum value deconvolved. On top: electro-physiological recordings, below: deconvolved activity from the imaging run. The frame rate used for capturing data is 17Hz[18]. Data from Akerboom *et al.*[18]. Note that the second to last row corresponds to the data depicted in Figure 2; the set consists of 9 annotated cells captured over 12000 frames.

2.2.2 Accuracy for measuring overall activity

As mentioned in 2.2.1, data with small, distinct burst of activity is the easiest to analyze, and the number of action potentials observed within one ROI/cell can be expected to be accurate.
More densely packed activity, consequently, increases the difficulty of gaging the amount of spikes.
Accordingly, one has to consider the data at hand when using CNI to look at the total amount of spikes in a cell, for the reliability of such an analysis is quite variable.
Generally speaking, deconvolution of CNI data tends to yield lower amounts of APs than electro-physiological methods, which is a well recorded behavior.[21][39]
It is important to keep the similarity of data in mind, however: comparing overall activity between cells of the same species and type, with less signal-type determining parameters changed (e.g. knockout or stimulus), is not only possible but can be expected to yield high accuracy.
For testing, I simulated 5 sets (of 20 ROIs with a recording length of 20000 frames) and a 10% total AP increase contained between sets, respectively.
The selected values for action potentials were chosen such that they assess the accuracy over different densities of data, ranging from a set with 25,28,31,34 and 37 APs per ROI, resulting in sparse and distinct transients, to heavily overlapping traces with 200,220,240,260 and 280 APs contained. See Figure 6.

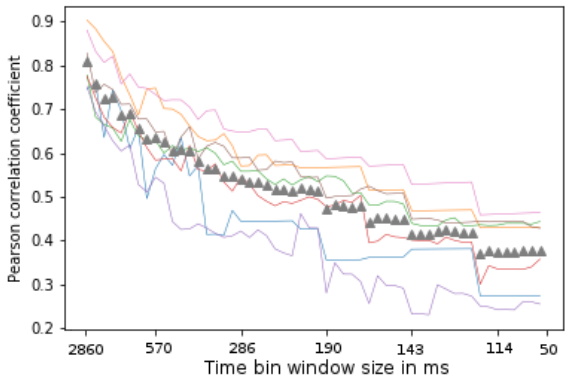


Figure 4: Correlations of the deconvolved spike trains with ones derived from patch-clamp recordings, which are the gold standard for AP detection. Due to the difficulties with achieving frame-wise precision mentioned in this section, direct comparison of the two traces generally yields very low scores and is barely informative[38]. Depicted here is thus the varying degree of Pearson-correlation w.r.t. binning AP activity over changing bin sizes. The left-most position on the x-axis depicts a total of 100 bins distributed over the traces (by Akerboom *et al.*[18]), corresponding to roughly a 2.8-second window, while the right-most coordinate shows 114ms. One trace corresponds to the varying correlation values w.r.t. bin size of one individual cell. As is to be expected, higher temporal scrutiny decreases the correlation of the two methods, whereas an increase in bin size reveals strong similarity over all sets of data. The mean score of all traces depicted is marked with grey triangles. Assuming a correlation coefficient of 0.4 to be satisfactory[38], reducing bins below a 50ms-window is not feasible, while the ranges above this value can be considered reliable. Note that some traces consistently exhibit better scores than others, which is likely due to a clearer distinction between transients and baseline regions, i.e. clearer data. The temporary drop in some regions of trace scores is likely due to local spikes in activity, which, at the corresponding position, are divided over several bins. The score subsequently rises above its previous level because the higher resolution partitions the local spikes over multiple bins, thus circumventing the problem.

All free parameters were left at default values for the tests, and each test repeated five times (that is, 5 sets of data were generated for each number of APs, totaling 5*5*5 = 125 sets with 20 ROIs each).
Two realizations of these simulated traces are shown in Figure 6, the corresponding accuracy assessments are displayed in the table below.

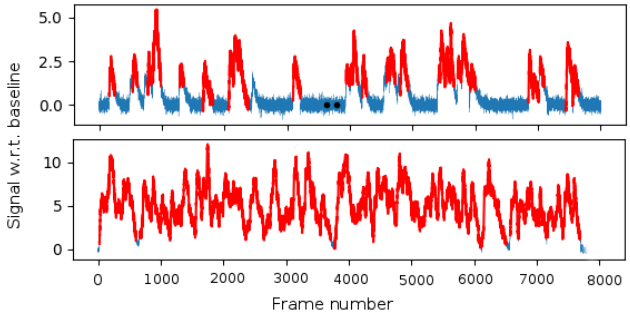


Figure 6: Top: simulation of sparse data, containing 20 APs. Bottom: simulation of dense data, featuring 280 APs. As before, regions labeled as transients are highlighted in red, with non-transient trace in blue. While the data on top is relatively easy to deconvolve, achieving good accuracy for the bottom one is hard.

Base set AP num	Total AP/Deconvolved AP	Ordering preserved
25	0.95(+/- 0.13)	5/5
50	1.14(+/- 0.08)	4/5
100	1.43(+/- 0.09)	2/5
150	1.87(+/- 0.11)	1/5
200	2.20(+/- 0.21)	0/5

The general behavior observed is that an increase in transient density, and according rise in overlap, causes the procedure to underestimate the number of action potentials in the trace.
This is because the permanent activity causes the calculation of the alpha kernel to be overly conservative. When all distinguishable reference slices of data contain multiple overlapping transients, it is increasingly harder to retrieve single-peaked references.
The data used for calculation of the deconvolution kernel consequently consists of multi-AP response traces, causing the method to produce a lower amount of total firing activity.

A 10% margin of activity between groups of cells can reliably be detected, given suitable data. Conversely, it should be kept in mind that less easily deconvolvable traces should be treated with care, by e.g. only considering bigger difference margins as indicative of an underlying change, or changing some of the algorithm's free parameters.
Repeating the same trial but with a 20% margin between 'runs' in-set yields the following:

Base set AP num	Total AP/Deconvolved AP	Ordering preserved
25	0.92(+/- 0.1)	5/5
50	0.91(+/- 0.01)	5/5
100	1.31(+/- 0.03)	4/5
150	1.63(+/- 0.05)	4/5
200	1.89(+/- 0.06)	2/5

As expected, increasing the AP margin between runs leads to a significant improvement for the ranking accuracy of comparing the sets, while it has little to no impact on the precision of the AP number deconvolved.
Sparse data yields better results when increasing the slope-cutoff for the event detection and window size considered for the calculation thereof.
The number of APs selected for accuracy assessments are by no means applicable to the method in general, but rather heavily reliant upon the underlying single-AP kernel. The model used here took parameters from transients observed in mouse mPFC pyramidal neurons; data featuring either shorter or longer bursts of fluorescence would increase or lower the optimal range for the technique.

2.2.3 Runtime

The algorithm runs in a single process. This means that launching the script multiple times with independent sets of data will cause the operating system to assign different processes automatically, executing them in parallel. This may not be true in the case of a distributed system, but holds for Unix-based operating systems as well as Windows.
A 100mb set of data, which corresponds to roughly 6 million total float values stored in UTF-8, distributed over multiple ROIs or cells, takes 2 minutes to complete, using an Intel i5 second generation processor with 4 2.60GHz CPUs and 4GB RAM. These system specifications are below an average desktop computer, and the script can be run once per core (depending on the operating system), meaning this time is quartered if the input is partitioned accordingly.
Performing the same test on the data used for accuracy assessments above, the

runtime for the 112 cells of about 53000 frames or 1180000 float values is 45.7 seconds total, or roughly 1100 frames per second. Since the data is naturally divided into 5 sets, it is possible to launch a separate process on each core, which further reduces the time required to completion.

2.3 Running the Script

The script is intended to be run from a terminal.
In order to invoke a python program, find the path to the corresponding file (main.py in this case), and start the call with "python main.py" - this tells your python interpreter to run the code contained in the file listed as the second argument.
Input files should be in CSV format(comma-separated values), with one floating point value corresponding to the average fluorescence of a cell in that frame. As elaborated in section **??**, this requires preceding cell annotation of the recordings, for which a multitude of algorithms are available.
The files can list cells vs. frames either row or column-based, and recording lengths do not have to contain identical frame numbers.[1] Data contained within one file is pooled for generation of the alpha kernel, whereas separate files fed into the algorithm simultaneously generate one kernel per file. Input should thus be compiled according to similarity, and combining different cell types, sampling rates or dyes in a file is not recommended. Combination of similar recordings, on the other hand, increases the amount of transients sampled for the kernel, and causes a higher reliability of the method.
A minimal example of the syntax for calling the tool via a terminal:

```
>> python /path/to/script/main.py InputFile
```

The script *main.py* is called with a preceding *python* statement. Multiple input files can be listed, and data contained within one file is pooled for calculation of the alpha kernel. The output directory specified is optional: if none is supplied at run-time, files are put in the directory the script is called from. Keep in mind that directory syntax as well as general string formatting issues depend on the operating system. The subsequent arguments should be one or several input files containing the data, followed by a path to the directory which the output should be written to, resulting in the following syntax:

```
>> python main.py File1 File2 /path/to/output/directory/
```

Depending on the operating system, file paths can either be relative to the current working directory (Unix/Windows), or have to be complete.
It should be noted that all parameters outside of the actual input files have

[1] Input of varying total recording length causes the shorter recordings to be padded with zeros. This is because the API utilizes algorithms optimized for matrix calculus, and thus requires arrays of the same length. A "divide by zero"-warning is thus printed when the script is used with such input; it does however not change the API's behavior.

default values, which is why the script can be called without specifying anything else.
Nonetheless, it might be advantageous to change some of the parameters of the analysis, with the most crucial ones being whether quantile correction (see section **??**) should be applied, as well as selecting a fitting window size for the baseline. The syntax for changing these variables[2] is:

$-q$	quantile bin size integer
$-b$	baseline bin size integer

Anything roughly in the range of one tenth of the total number of frames works, unless the data exhibits permanent activity and no clear baseline of that size can be determined - in which case the deconvolution in its entirety is very hard, and results have to be rigorously checked.
As elaborated in section **??**, the bin considered for quantile normalization should not be too small, or resulting suppression of transients is to be expected. The default for this parameter is 0, no quantile normalization is done unless specified otherwise. This is because it is only necessary in order to correct for drift, and should not be applied unless necessary.
Should that be the case, supplying data slices ranging from one sixth to one third of the total time series should suffice in order to remove drift from the data.
Other free variables are the number of adjacent frames considered for slope calculations, the amount of consecutive threshold hits required for a transient onset to be declared, the minimum length (in frames) a transient should exhibit, and the distance from the slope distribution mean at which the cutoff for transient detection is located (in standard deviances). The triggers for runtime modification of these are:

$-s$	distance between frames considered for the slope calculation(integer)
$-c$	slope threshold location relative to slope distribution mean(float)
$-m$	minium number of data points above transient start fluorescence(integer)
$-n$	number of consecutive frames required to be above the baseline(integer)

All of these have default values which worked well for mouse visual cortex mPFC data, but can be adjusted via command-line at will.
Two additional parameters, -o and -s, are used to specify where to store the output and which character is used to delimit(separate) values in the input file.
Regarding a bigger distance between data points for calculation of slopes causes lower susceptibility to noise, at the cost of decreased sensitivity. Bigger margins flatten the slope values, and are comparable to a floating average in their behavior. The minimum length (-n) for transients behaves in a similar way; longer transient requirements remove short stretches of data from the list compiled for

[2]The default baseline argument looks for a a 300 frame window to calculate the baseline activity.

ongoing analysis. This also applies to an increase of the -m parameter, which determines the amount of consecutive slope points above threshold required in order to declare a transient onset.
Consequently, the n,m and s parameters have default values in their lower range, where they do not hinder sensitivity. If not changed during runtime, slopes(-s) are calculated directly from neighboring values, transients shorter than 10 frames are discarded (-n) and 3 consecutive slope values have to be above the threshold (-m).
The parameter changed with -c behaves slightly differently. It determines the exact value of the threshold, which is calculated as the standard deviance of the distribution, explained in Figure **??**, times this parameter. Unchanged, it is defaulted to 2, lowering it allows less steep onset slopes to be labeled as transients, while higher values do the opposite.
Here is an example for the syntax used when changing multiple parameters:

```
>> python main.py InputFile -q 1500 -m 10 -n 5 -c 2 -s 3
```

Notice that their ordering is arbitrary, but the input file/s are obligatory and have to precede any other statements, placing them right behind the script name *main.py*.
If the syntax used to call the script is incorrect, it prints a brief explanation of how it is supposed to be invoked to the terminal and exits.

2.4 CIDA Main Components

The command-line tool explained above is little more than a top-down list of functions contained within the "cal_neuroim"-module and argument parsing / file reading routines.
All of the methods are thus intended to be used for writing new analysis programs. The fastest way to achieve this is putting the file "cal_neuroim.py" in the same directory as the new script and calling its functions directly, or adding the module to the python directory.
In order to get a general idea as to how the functions are called, let us take the code from "main.py"(the script) as an example.
It should generally be noted that methods starting with an underscore (_) are being called from higher-level functions and were generally not intended to be used on their own.

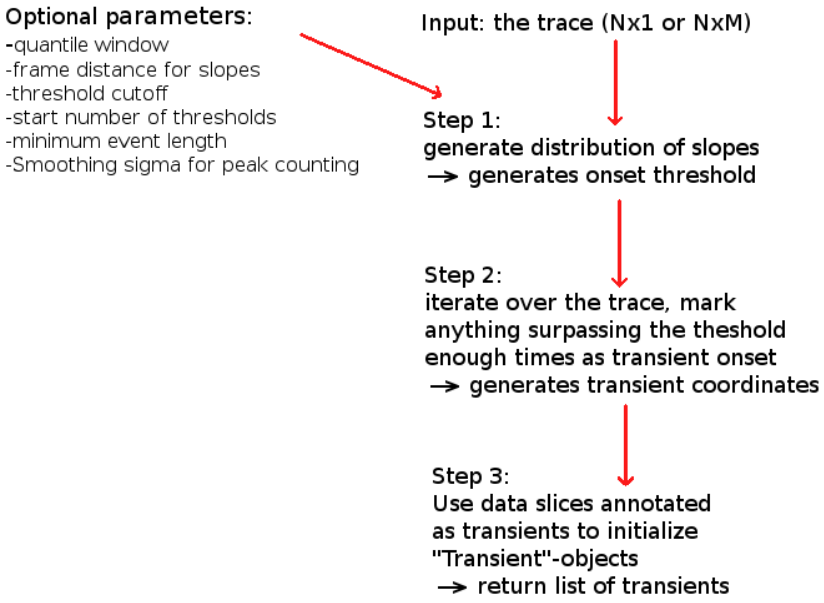
2.4.1 importMatrix

This function reads the file located at the supplied path, which should contain a time series' raw values column-wise.
The second argument tells the function what separator is used to distinguish entries if the file is in csv-format (human-readable encoding), or treats it as an xlrd file if no second argument is supplied or it is null.
It checks if all input recordings are of the same length, and pads the shorter ones

with zeros if this is not the case. The function was written for convenience's sake, and it turns a time-series file into a numpy ndarray object. If your data already fulfills this condition, importMatrix is not needed: File imports are generally very straight-forward in python.

2.4.2 eventDetect

This method is the core piece of the API, and its performance is critical for robust and reliable analysis.
If supplied with a non-zero quantile width argument, eventDetect performs quantile normalization at the same time as transient annotation; which saves an iteration over the data.
If quantile normalization should be done separately, an additional function called quantileNorm is available in the script. Most of the free parameters (s,c,m,n,q)

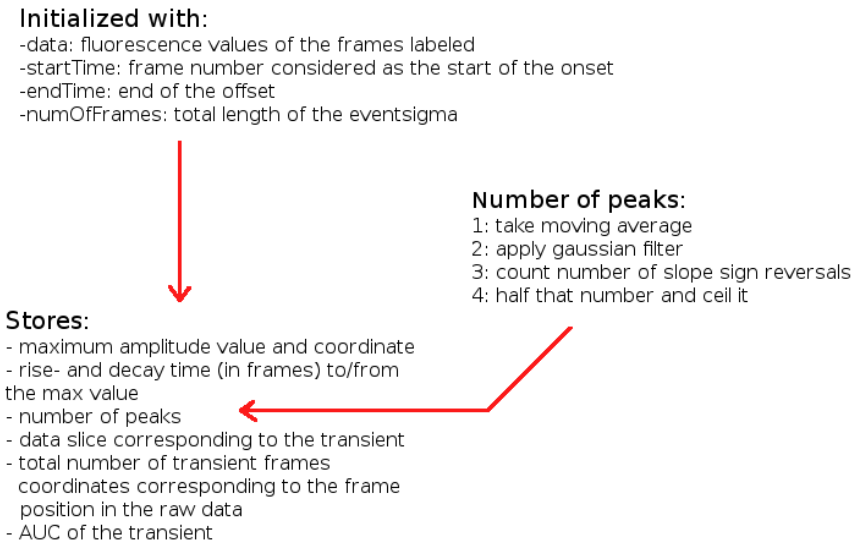


that can be adapted at run-time are for this function. Depending on their values, the algorithm moves along the specificity- sensitivity trade-off axis, by enforcing increasingly strong requirements for a slice of data to be considered a transient.
The return value is a tuple containing 3 items: transients, a matrix of the quantile normalized values and the distribution of slopes calculated from each time series.
The last two are not used for any further analysis, but rather serve for visualization and checking whether the corrections were appropriate and the slope threshold reliable. The first item returned is the list, or list of lists, depending on the input dimension, of transients.

An alternate procedure is contained in the library, which uses the thresholding method explained in section **??**; it is called "thresholdEventDetect". It takes less optional parameters due to the thresholding procedure being more straightforward in its approach. It also accepts a cutoff-parameter and a minimum transient length requirement, as well as a quantile window if the normalization should be performed. The return values are the same as for the slope-dependent method, minus the slope distribution.

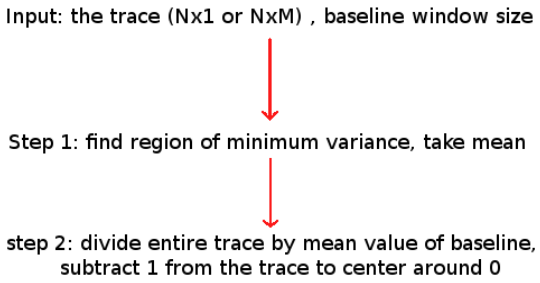
2.4.3 The Transient Class

While the class itself is called "_Transient", and accordingly not intended to be called directly, the eventDetect method explained above returns a list of lists of objects of this class.
Upon being flagged as a region of interest by the event detection, the corresponding slice of data is passed to initialize a transient-object.
Each instance of these contains the following properties:
Start and end time, relative to the data (i.e. the frames determined to be the start and end of the activity), amplitude, rise and decay time, number of peaks, the normalized data constituting the transient and the total number of frames it lasted.
Keep in mind however that eventDetect returns a list of lists of these transient objects - one list of transients per cell, which are themselves each an entry in a list of the cells. So passing 4 traces, or cells, to the method would yield 4 lists of variable lengths: The number of transients contained within the recorded cells are not expected to be identical, or even similar.



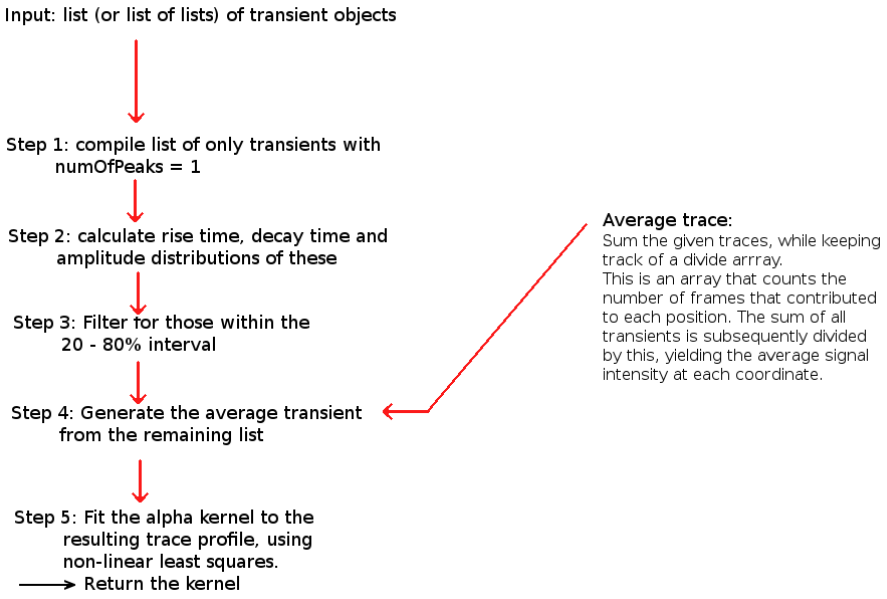
2.4.4 pushToBaseline

As the name implies, this is the method for baseline correction of the data. Contained within are calls to several sub-functions also found in the library, such as the identification routine of the region to be used as the reference baseline. It takes the data to be normalized and the length of the window which is used to determine the baseline. It returns a tuple of the normalized data matrix and a list of coordinate tuples, which specify the first and last frame number of the region used as baseline. The first element of the output tuple is thus the NxM or Nx1 matrix of data, and a list of coordinate tuples of length M.
See section **??** for reference.



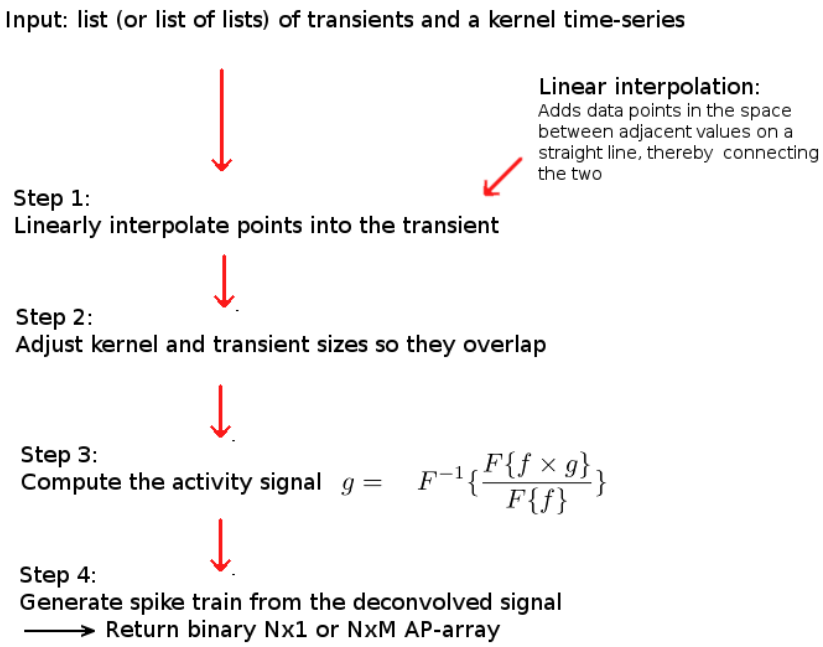
2.4.5 createMeanKernel

This method is invoked with a list (or list of lists) of transients, as returned by eventDetect or its thresholding counterpart.
It subsequently iterates over the number of peaks parameter stored in the transients, and compiles a list of those that are single-peaked. This list is then used to calculate the respective amplitude, rise and decay time distributions.
These ranges are needed to compute the "mean transient", as explained in section **??**.
The resulting time series is fitted to the alpha function, which is the methods return value: An ndArray containing the alpha kernel, of the size of the number of frames contained in the data.
Because the alpha kernel is as long as the longest transient contained in the template list, which in turn is composed of non-outlier transients, there are cases where the kernel array is shorter than the data slice to be deconvolved. The function automatically extends the kernel in this case.



2.4.6 Deconvolve and _generateSpiketrainFromSignal

Deconvolve requires a list (or list of lists) of transients and a kernel time series (as created above) to be used for the deconvolution.
It then applies the convolution theorem (section **??**) to extract the spiking signal.
The function's output is an ndArray initialized with zeros; the data points which previously held activity are replaced by the deconvolved signal.
The script then passes the derived activity measure directly to _generateSpiketrainFromSignal, which determines how many spikes to assign to which frames; this algorithm is further described in section **??**.
The final output, used for generating plots in the "main.py"-script, is a binary {0,1} array depicting the presence of spikes in a frame-wise manner, each row corresponding to one ROI of the initial input.



3 Conclusion

CIDA performs well as long as a suitable reference can be found, which is the case for non-continuous spiking activity. Data which does not allow this distinction to be made is generally hard to deconvolve, however, and it is questionable whether more complex algorithms can tackle this problem.
CNI itself is not aiming at high temporal or spiking resolutions, for which patch-clamps are the gold standard[52]. Its strengths are rather the usability as well as the high number of cells captured per run[20], which high-complexity algorithms with explosive computational cost do not remedy. While other analysis procedures yield higher accuracy scores, straight-forward deconvolution allows for the same conclusions to be drawn from data: the properties of CNI do not allow it to assess minute spiking fluctuations; regardless of small gains in correlation.
Since algebraic deconvolution is highly dependent on an adequate selection of template transients for its kernel, upping the algorithmic complexity slightly in favor of a more dedicated annotation process improves its performance on both simulated and supervised in-vivo data, while remaining very fast comparatively.
Additionally, most, if not all, published methods for analyzing CNI data are either implemented in matlab or do not provide any code or implementation details[41][42][43][44][45][46][47].
The API presented here, on the other hand, is freely available via a GNU public license, thus allowing for at-will modification of the code, has no further depen-

dencies outside of standard python libraries, and includes a script which can be run via console to deconvolve supplied data.
Furthermore, it can feasibly be extended to an on-line method, enabling analysis of experiments as they are being conducted, thus allowing for ongoing development going forward.

4 Acknowledgements

I would like to express my gratitude to Marie Rooy and Dr. Boris Gutkin of the ENS Paris, who helped me with great patience through my numerous misunderstandings and my initial lack of knowledge concerning the subject.
Several people helped me by providing feedback and pointing out mistakes in the work, first and foremost Claire Cooper, without whom this thesis surely would have suffered from an even greater abundance of flaws.
Without the aid of these people, this work would not have been possible.
Thank you.

References

1. Dayan, P. & Abbott, L. F. *Theoretical neuroscience* (Cambridge, MA: MIT Press, 2001).
2. Pachitariu, M. *et al.* in *Advances in Neural Information Processing Systems 26* (eds Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z. & Weinberger, K. Q.) 1745–1753 (Curran Associates, Inc., 2013). <`http://papers.nips.cc/paper/5167-extracting-regions-of-interest-from-biological-images-with-convolutional-sparse-block-coding.pdf`> (visited on 06/19/2017).
3. Denk, W., Strickler, J. H. & Webb, W. W. Two-photon laser scanning fluorescence microscopy. eng. *Science (New York, N.Y.)* **248,** 73–76. ISSN: 0036-8075 (Apr. 1990).
4. Dombeck, D. A., Khabbaz, A. N., Collman, F., Adelman, T. L. & Tank, D. W. Imaging Large-Scale Neural Activity with Cellular Resolution in Awake, Mobile Mice. English. *Neuron* **56,** 43–57. ISSN: 0896-6273 (Oct. 2007).
5. Yaksi, E. & Friedrich, R. W. Reconstruction of firing rate changes across neuronal populations by temporally deconvolved Ca2+ imaging. en. *Nature Methods* **3,** 377–383. ISSN: 1548-7091 (May 2006).
6. Smith, S. L. & Häusser, M. Parallel processing of visual space by neighboring neurons in mouse visual cortex. en. *Nature Neuroscience* **13,** 1144–1149. ISSN: 1097-6256 (Sept. 2010).
7. Mukamel, E. A., Nimmerjahn, A. & Schnitzer, M. J. Automated analysis of cellular signals from large-scale calcium imaging data. eng. *Neuron* **63,** 747–760. ISSN: 1097-4199 (Sept. 2009).
8. Maruyama, R. *et al.* Detecting cells using non-negative matrix factorization on calcium imaging data. *Neural Networks* **55,** 11–19. ISSN: 0893-6080 (July 2014).
9. Tomek, J., Novak, O. & Syka, J. Two-Photon Processor and SeNeCA: a freely available software package to process data from two-photon calcium imaging at speeds down to several milliseconds per frame. eng. *Journal of Neurophysiology* **110,** 243–256. ISSN: 1522-1598 (July 2013).
10. Schmidt, R. F., Lang, F. & Heckmann, M. *Physiologie des Menschen: mit Pathophysiologie* 30th ed. Deutsch. ISBN: 978-3-540-32908-4 (Springer, Heidelberg, Sept. 2007).
11. Löffler, G. & Petrides, P. E. *Biochemie und Pathobiochemie* 6th ed. Deutsch. ISBN: 978-3-540-64350-0 (Springer, Berlin, Sept. 1998).
12. Purves, D. *et al.* Ion Channels Underlying Action Potentials. en. <`https://www.ncbi.nlm.nih.gov/books/NBK10841/`> (visited on 10/17/2017) (2001).

13. Bahar, E., Kim, H. & Yoon, H. ER Stress-Mediated Signaling: Action Potential and Ca2+ as Key Players. en. *International Journal of Molecular Sciences* **17,** 1558 (Sept. 2016).

14. Nover, L. *et al. Lehrbuch der Molekularen Zellbiologie* 3. vollständig überarbeitete Auflage. Deutsch. ISBN: 978-3-527-31160-6 (Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, Apr. 2005).

15. Dombeck, D. & Tank, D. Two-photon imaging of neural activity in awake mobile mice. eng. *Cold Spring Harbor Protocols* **2014,** 726–736. ISSN: 1559-6095 (July 2014).

16. Miyazaki, K. & Ross, W. N. Simultaneous Sodium and Calcium Imaging from Dendrites and Axons. *eNeuro* **2.** ISSN: 2373-2822. doi:10.1523/ENEURO.0092-15.2015. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4699831/> (visited on 10/13/2017) (Nov. 2015).

17. Dana, H. *et al.* Sensitive red protein calcium indicators for imaging neural activity. en. *eLife* **5,** e12727. ISSN: 2050-084X (Mar. 2016).

18. Akerboom, J. *et al.* Optimization of a GCaMP calcium indicator for neural activity imaging. eng. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience* **32,** 13819–13840. ISSN: 1529-2401 (Oct. 2012).

19. Chen, T.-W. *et al.* Ultrasensitive fluorescent proteins for imaging neuronal activity. eng. *Nature* **499,** 295–300. ISSN: 1476-4687 (July 2013).

20. Grienberger, C. & Konnerth, A. Imaging calcium in neurons. eng. *Neuron* **73,** 862–885. ISSN: 1097-4199 (Mar. 2012).

21. Orbach, H. S., Cohen, L. B. & Grinvald, A. Optical mapping of electrical activity in rat somatosensory and visual cortex. en. *Journal of Neuroscience* **5,** 1886–1895. ISSN: 0270-6474, 1529-2401 (July 1985).

22. Li, M., Liu, F., Jiang, H., Lee, T. S. & Tang, S. Long-Term Two-Photon Imaging in Awake Macaque Monkey. English. *Neuron* **93,** 1049–1057.e3. ISSN: 0896-6273 (Mar. 2017).

23. Tian, L. *et al.* Imaging neural activity in worms, flies and mice with improved GCaMP calcium indicators. en. *Nature Methods* **6,** 875–881. ISSN: 1548-7091 (Dec. 2009).

24. Pnevmatikakis, E. A. *et al.* Simultaneous Denoising, Deconvolution, and Demixing of Calcium Imaging Data. English. *Neuron* **89,** 285–299. ISSN: 0896-6273 (Jan. 2016).

25. *Meaney Lab at the University of Pennsylvania - People* <http://www.seas.upenn.edu/~molneuro/projects.html> (visited on 10/17/2017).

26. Kaifosh, P., Zaremba, J. D., Danielson, N. B. & Losonczy, A. SIMA: Python software for analysis of dynamic fluorescence imaging data. English. *Frontiers in Neuroinformatics* **8.** ISSN: 1662-5196. doi:10.3389/fninf.2014.00080. <http://journal.frontiersin.org/article/10.3389/fninf.2014.00080/full> (visited on 06/19/2017) (2014).

27. *The ImageJ ecosystem: An open platform for biomedical image analysis - Schindelin - 2015 - Molecular Reproduction and Development - Wiley Online Library* <http://onlinelibrary.wiley.com/doi/10.1002/mrd.22489/full> (visited on 06/19/2017).
28. *Conda — Conda documentation* <https://conda.io/docs/> (visited on 11/29/2017).
29. Smetters, D., Majewska, A. & Yuste, R. Detecting Action Potentials in Neuronal Populations with Calcium Imaging. *Methods* **18,** 215–221. ISSN: 1046-2023 (June 1999).
30. Arndt, J. & Haenel, C. *Pi: Algorithmen, Computer, Arithmetik* de. Google-Books-ID: jQgmBgAAQBAJ. ISBN: 978-3-662-09360-3 (Springer-Verlag, Mar. 2013).
31. Krystek, M. *Die digitale Implementierung des Profilfilters nach DIN EN ISO 11562* de. Google-Books-ID: ITqZNgle4xUC. ISBN: 978-3-410-15861-5 (Beuth Verlag, 2004).
32. *GNU General Public License* <http://www.gnu.org/licenses/gpl.html> (Free Software Foundation, June 2007).
33. Oliphant, T. E. Python for Scientific Computing. *Computing in Science Engineering* **9,** 10–20. ISSN: 1521-9615 (May 2007).
34. *pandas: a Foundational Python Library for Data Analysis and Statistics | R (Programming Language) | Database Index* <https://www.scribd.com/document/71048089/pandas-a-Foundational-Python-Library-for-Data-Analysis-and-Statistics> (visited on 06/15/2017).
35. Walt, S. v. d., Colbert, S. C. & Varoquaux, G. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science Engineering* **13,** 22–30. ISSN: 1521-9615 (Mar. 2011).
36. *Welcome to the CRCNS data sharing website — CRCNS.org* <http://crcns.org/> (visited on 07/21/2017).
37. Teeters, J. L., Harris, K. D., Millman, K. J., Olshausen, B. A. & Sommer, F. T. Data Sharing for Computational Neuroscience. en. *Neuroinformatics* **6,** 47–55. ISSN: 1539-2791, 1559-0089 (Mar. 2008).
38. Theis, L. *et al.* Benchmarking Spike Rate Inference in Population Calcium Imaging. *Neuron* **90,** 471–482. ISSN: 0896-6273 (May 2016).
39. Hofer, S. B. *et al.* Differential connectivity and response dynamics of excitatory and inhibitory neurons in visual cortex. en. *Nature Neuroscience* **14,** 1045–1052. ISSN: 1097-6256 (Aug. 2011).
40. Friedrich, J., Zhou, P. & Paninski, L. Fast online deconvolution of calcium imaging data. *PLOS Computational Biology* **13,** 1–26 (2017).
41. Ranganathan, G. N. & Koester, H. J. Optical Recording of Neuronal Spiking Activity From Unbiased Populations of Neurons With High Spike Detection Efficiency and High Temporal Precision. en. *Journal of Neurophysiology* **104,** 1812–1824. ISSN: 0022-3077, 1522-1598 (Sept. 2010).

42. Sasaki, T., Takahashi, N., Matsuki, N. & Ikegaya, Y. Fast and Accurate Detection of Action Potentials From Somatic Calcium Fluctuations. en. *Journal of Neurophysiology* **100,** 1668–1676. ISSN: 0022-3077, 1522-1598 (Sept. 2008).

43. Vogelstein, J. T. *et al.* Spike inference from calcium imaging using sequential Monte Carlo methods. eng. *Biophysical Journal* **97,** 636–655. ISSN: 1542-0086 (July 2009).

44. Deneux, T. *et al.* Accurate spike estimation from noisy calcium signals for ultrafast three-dimensional imaging of large neuronal populations *in vivo*. en. *Nature Communications* **7,** ncomms12190. ISSN: 2041-1723 (July 2016).

45. Kerr, J. N. D., Greenberg, D. & Helmchen, F. Imaging input and output of neocortical networks in vivo. *Proceedings of the National Academy of Sciences of the United States of America* **102,** 14063–14068. ISSN: 0027-8424 (Sept. 2005).

46. Ozden, I., Lee, H. M., Sullivan, M. R. & Wang, S. S.-H. Identification and Clustering of Event Patterns From In Vivo Multiphoton Optical Recordings of Neuronal Ensembles. *Journal of Neurophysiology* **100,** 495–503. ISSN: 0022-3077 (July 2008).

47. Romano, S. A. *et al.* An integrated calcium imaging processing toolbox for the analysis of neuronal population dynamics. *PLOS Computational Biology* **13,** e1005526. ISSN: 1553-7358 (June 2017).

48. Agarwal, A. *Computational Trade-offs in Statistical Learning | EECS at UC Berkeley* 2012. <https://www2.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-169.html> (visited on 11/17/2017).

49. *New License for MATLAB R2017b - MathWorks Deutschland* <https://de.mathworks.com/store/link/products/standard/ML> (visited on 11/18/2017).

50. *MATLAB-Lizenz zur campusweiten Verwendung* <https://de.mathworks.com/academia/matlab-campus.html> (visited on 11/18/2017).

51. *PerformancePython - SciPy wiki dump* <http://scipy.github.io/old-wiki/pages/PerformancePython> (visited on 11/18/2017).

52. Yajuan, X., Xin, L. & Zhiyuan, L. A Comparison of the Performance and Application Differences Between Manual and Automated Patch-Clamp Techniques. *Current Chemical Genomics* **6,** 87–92. ISSN: 1875-3973 (Dec. 2012).