PLOS | COMPUTATIONAL BIOLOGY

RESEARCH ARTICLE

Fast online deconvolution of calcium imaging data

Johannes Friedrich[1,2]*, **Pengcheng Zhou**[1,3], **Liam Paninski**[1,4]

1 Department of Statistics, Grossman Center for the Statistics of Mind, and Center for Theoretical Neuroscience, Columbia University, New York, New York, United States of America, **2** Janelia Research Campus, Ashburn, Virginia, United States of America, **3** Center for the Neural Basis of Cognition and Machine Learning Department, Carnegie Mellon University, Pittsburgh, Pennsylvania, United States of America, **4** Kavli Institute for Brain Science, and NeuroTechnology Center, Columbia University, New York, New York, United States of America

* j.friedrich@columbia.edu

Citation: Friedrich J, Zhou P, Paninski L (2017) Fast online deconvolution of calcium imaging data. PLoS Comput Biol 13(3): e1005423. https://doi.org/10.1371/journal.pcbi.1005423

Editor: Joshua Vogelstein, Johns Hopkins University, UNITED STATES

Received: November 7, 2016

Accepted: February 24, 2017

Published: March 14, 2017

Data Availability Statement: All code is available on https://github.com/j-friedrich/OASIS and linked repositories therein. We used a publicly available dataset provided by the GENIE project, Svoboda lab, at Janelia Research Campus on http://crcns.org (Chen et al., 2013; GENIE project, 2015).

Funding: Funding for this research was provided by Swiss National Science Foundation (http://www.snf.ch) Research Award P300P2_158428 (JF), National Institutes of Health (NIH, https://www.nih.gov) 2R01MH064537 and R90DA023426 (PZ), Simons Foundation (https://www.simonsfoundation.org) Global Brain Research

Abstract

Fluorescent calcium indicators are a popular means for observing the spiking activity of large neuronal populations, but extracting the activity of each neuron from raw fluorescence calcium imaging data is a nontrivial problem. We present a fast online active set method to solve this sparse non-negative deconvolution problem. Importantly, the algorithm progresses through each time series sequentially from beginning to end, thus enabling real-time online estimation of neural activity during the imaging session. Our algorithm is a generalization of the pool adjacent violators algorithm (PAVA) for isotonic regression and inherits its linear-time computational complexity. We gain remarkable increases in processing speed: more than one order of magnitude compared to currently employed state of the art convex solvers relying on interior point methods. Unlike these approaches, our method can exploit warm starts; therefore optimizing model hyperparameters only requires a handful of passes through the data. A minor modification can further improve the quality of activity inference by imposing a constraint on the minimum spike size. The algorithm enables real-time simultaneous deconvolution of $O(10^5)$ traces of whole-brain larval zebrafish imaging data on a laptop.

Author summary

Calcium imaging methods enable simultaneous measurement of the activity of thousands of neighboring neurons, but come with major caveats: the slow decay of the fluorescence signal compared to the time course of the underlying neural activity, limitations in signal quality, and the large scale of the data all complicate the goal of efficiently extracting accurate estimates of neural activity from the observed video data. Further, current activity extraction methods are typically applied to imaging data after the experiment is complete. However, in many cases we would prefer to run closed-loop experiments—analyzing data on-the-fly to guide the next experimental steps or to control feedback—and this requires new methods for accurate real-time processing. Here we present a fast activity extraction

Awards 325171 and 365002 (JF,LP), Army Research Office (ARO, https://www.arl.army.mil?page=472) MURI W911NF-12-1-0594, NIH BRAIN Initiative (https://www.braininitiative.nih.gov) R01 EB22913 and R21 EY027592, Defense Advanced Research Projects Agency (DARPA, http://www.darpa.mil) N66001-15-C-4032 (SIMPLEX), and a Google Faculty Research award (LP); in addition, this work was supported by the Intelligence Advanced Research Projects Activity (IARPA, https://www.iarpa.gov) via Department of Interior/ Interior Business Center (DoI/IBC) contract number D16PC00003, D16PC00008 (LP) and D16PC00007 (PZ). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/IBC, or the U.S. Government.

Competing interests: The authors have declared that no competing interests exist.

algorithm addressing both issues. Our approach follows previous work in casting the activity extraction problem as a sparse nonnegative deconvolution problem. To solve this optimization problem, we introduce a new algorithm that is an order of magnitude faster than previous methods, and progresses through the data sequentially from beginning to end, thus enabling, in principle, real-time online estimation of neural activity during the imaging session. This computational advance thus opens the door to new closed-loop experiments.

This is a *PLOS Computational Biology* Methods paper.

Introduction

Calcium imaging has become one of the most widely used techniques for recording activity from neural populations in vivo [1]. The basic principle of calcium imaging is that neural action potentials (or spikes), the point process signal of interest, each induce an optically measurable transient response in calcium dynamics. The nontrivial problem of extracting the activity of each neuron from a raw fluorescence trace has been addressed with several different approaches, including template matching [2] and linear deconvolution [3, 4], which are outperformed by sparse non-negative deconvolution [5]. The latter can be interpreted as the maximum a posteriori (MAP) estimate under a simple generative model (linear convolution plus noise; Fig 1), whereas fully Bayesian methods [6–8] can provide some further improvements, but are more computationally expensive. Supervised methods trained on simultaneously-recorded electrophysiological and imaging data [9, 10] have also recently achieved state of the art results, but are more black-box in nature; Bayesian methods based on a well-defined generative model are somewhat easier to generalize to more complex multi-neuronal or multi-trial settings [11–13].

The methods above are typically applied to imaging data offline, after the experiment is complete; however, there is a need for accurate and fast real-time processing to enable closed-loop experiments, a powerful strategy for causal investigation of neural circuitry [14]. In particular, observing and feeding back the effects of circuit interventions on physiologically relevant timescales will be valuable for directly testing whether inferred models of dynamics, connectivity, and causation are accurate in vivo, and recent experimental advances [15, 16] are now enabling work in this direction. Brain-computer interfaces (BCIs) also rely on real-time estimates of neural activity. Whereas most BCI systems rely on electrical recordings, BCIs have

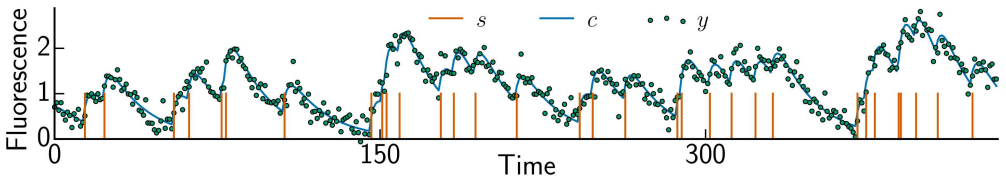

Fig 1. Generative autoregessive model for calcium dynamics. Spike train ***s*** gets filtered to produce calcium trace ***c***; here we used $p = 2$ as order of the AR process. Added noise yields the observed fluorescence ***y***.

https://doi.org/10.1371/journal.pcbi.1005423.g001

been driven by optical signals too [17], providing new insight into how neurons change their activity during learning on a finer spatial scale than possible with intracortical electrodes. Finally, adaptive experimental design approaches [18–20] also rely on online estimates of neural activity.

Even in cases where we do not require the strict timing/latency constraints of real-time processing, we still need methods that scale to large data sets as for example in whole-brain imaging of larval zebrafish [21, 22]. A further demand for scalability stems from the fact that the deconvolution problem is solved in the inner loop of constrained non-negative matrix factorization (CNMF) [13], the current state of the art for simultaneous denoising, deconvolution, and demixing of spatiotemporal calcium imaging data.

In this paper we address the pressing need for scalable online spike inference methods. Building on previous work, we frame this estimation problem as a sparse non-negative deconvolution. Current algorithms employ interior point methods to solve the ensuing optimization problem and are fast enough to process hundreds of neurons in about the same time as the recording [5], but can not handle larger data sets such as whole-brain zebrafish imaging in real time. Furthermore, these interior point methods scale linearly in the length of the recording, but they cannot be warm started [23], i.e., initialized with the solution from a previous iteration to gain speed-ups, and do not run online.

Here we note a close connection between the MAP problem and isotonic regression, which fits data by a monotone piecewise constant function. A classic algorithm for isotonic regression is the pool adjacent violators algorithm (PAVA) [24, 25], which can be understood as an online active-set optimization method. We generalized PAVA to derive an Online Active Set method to Infer Spikes (OASIS); this new approach to solve the MAP problem yields speed-ups in processing time by at least one order of magnitude compared to interior point methods on both simulated and real data. Further, OASIS can be warm-started, which is useful in the inner loop of CNMF, and also when adjusting model hyperparameters, as we show below. Importantly, OASIS is not only much faster, but operates in an online fashion, progressing through the fluorescence time series sequentially from beginning to end. The advances in speed paired with the inherently online fashion of the algorithm enable true real-time online spike inference during the imaging session (once the spatial shapes of neurons in the field of view have been identified), with the potential to significantly impact experimental paradigms.

Methods

This section is organized as follows. The first subsection introduces the autoregressive (AR(p)) model for calcium dynamics.

In the second subsection we derive an Online Active Set method to Infer Spikes (OASIS) for an AR(1) model. The algorithm is inspired by the pool adjacent violators algorithm (PAVA, Alg 1), which we review first and then generalize to obtain OASIS (Alg 2). This algorithm requires some hyperparameter values; the optimization of these hyperparameters is described next, along with several computational tricks for speeding up the hyperparameter estimation. We finally discuss thresholding approaches for reducing the number of small values returned by the original L1-penalized approach. The resulting problem is non-convex, and so we lose guarantees on finding global optima, but we can easily adapt OASIS to quickly find good solutions.

In the third subsection we generalize to AR(p) models of the calcium dynamics and describe a dual active set algorithm that is analogous to the one presented for the AR(1) case (Alg 2). However, this algorithm is greedy if $p > 1$ and yields only a good approximate

solution. We can refine this solution and obtain the exact result by warm-starting an alternative primal active set method we call ONNLS (Alg 3). Finally, Alg 4 summarizes all of these steps.

Model for calcium dynamics

We assume we observe the fluorescence signal for T timesteps, and denote by s_t the number of spikes that the neuron fired at the t-th timestep, $t = 1, \ldots, T$, cf. Fig 1. Following [5, 13], we approximate the calcium concentration dynamics $\boldsymbol{c}$ using a stable autoregressive process of order p (AR(p)) where p is a small positive integer, usually $p = 1$ or 2,

$$c_t = \sum_{i=1}^{p} \gamma_i c_{t-i} + s_t. \tag{1}$$

The observed fluorescence $\boldsymbol{y} \in \mathbb{R}^T$ is related to the calcium concentration as [5–7]:

$$y_t = a\, c_t + b + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2) \tag{2}$$

where a is a non-negative scalar, b is a scalar offset parameter, and the noise is assumed to be i.i.d. zero mean Gaussian with variance σ^2. For the remainder we assume units such that $a = 1$ without loss of generality. We begin by assuming $b = 0$ for simplicity, but we will relax this assumption later. (We also assume throughout that all parameters in sight are fixed; in case of e.g. drifting baselines b we could generalize the algorithms discussed here to operate over shorter temporal windows, but we do not pursue this here.) The parameters γ_i and σ can be estimated from the autocovariance function and the power spectral density (PSD) of $\boldsymbol{y}$ respectively [13]. The autocovariance approach assumes that the spiking signal $\boldsymbol{s}$ comes from a homogeneous Poisson process and in practice often gives a crude estimate of γ_i. We will improve on this below by fitting the AR coefficients directly, which leads to better estimates, particularly when the spikes have some significant autocorrelation.

The goal of calcium deconvolution is to extract an estimate $\hat{\boldsymbol{s}}$ of the neural activity $\boldsymbol{s}$ from the vector of observations $\boldsymbol{y}$. As discussed in [5, 13], this leads to the following non-negative LASSO problem for estimating the calcium concentration:

$$\underset{\hat{\boldsymbol{c}}, \hat{\boldsymbol{s}}}{\text{minimize}} \quad \tfrac{1}{2} \| \hat{\boldsymbol{c}} - \boldsymbol{y} \|^2 + \lambda \| \hat{\boldsymbol{s}} \|_1 \quad \text{subject to} \quad \hat{\boldsymbol{s}} = G\hat{\boldsymbol{c}} \geq 0 \tag{3}$$

where the ℓ_1 penalty on $\hat{\boldsymbol{s}}$ enforces sparsity of the neural activity and the lower triangular matrix G is defined as:

$$G = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ -\gamma_1 & 1 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ -\gamma_p & \cdots & -\gamma_1 & 1 & 0 & \cdots & 0 \\ 0 & -\gamma_p & \cdots & -\gamma_1 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -\gamma_p & \cdots & -\gamma_1 & 1 \end{pmatrix}. \tag{4}$$

The deconvolution matrix G is banded with bandwidth p for an AR(p) process. Equivalently, $\boldsymbol{s} = \boldsymbol{c} * \boldsymbol{g}$ with $\boldsymbol{g}$ a finite impulse response filter of order p ($p + 1$ filter taps) and $*$ denoting

convolution. To produce calcium trace $\boldsymbol{c}$, spike train $\boldsymbol{s}$ is filtered with the inverse filter of $\boldsymbol{g}$, an infinite impulse response $\boldsymbol{h}$, $\boldsymbol{c} = \boldsymbol{s} * \boldsymbol{h}$. (Although our main focus is on the autoregressive model, we will discuss more general convolutional observation models below as well, and touch on nonlinear effects such as saturation in the Appendix.) Following the approach in [5], note that the spike signal $\hat{\boldsymbol{s}}$ is relaxed from non-negative integers to arbitrary non-negative values.

Derivation of the active set algorithm

The optimization problem (3) could be solved using generic convex program solvers. Here we derive the much faster Online Active Set method to Infer Spikes (OASIS). The algorithm is inspired by the pool adjacent violators algorithm (PAVA) [24, 25], which we review first for readers not familiar with this classic algorithm before generalizing it to the non-negative LASSO problem.

Pool Adjacent Violators Algorithm (PAVA). The pool adjacent violators algorithm (Alg 1) is a classic exact algorithm for isotonic regression, which fits data by a non-decreasing piecewise constant function. This algorithm is due to [24] and was independently discovered by other authors [26, 27] as reviewed in [25, 28]. It can be considered as a dual active set method [29]. Formally, the (convex) problem is to

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x} - \boldsymbol{y}\|^2 \quad \text{subject to} \quad x_1 \leq \ldots \leq x_T. \tag{5}$$

We first present the algorithm in a way that conveys its core ideas (see Alg A in S1 Appendix), then improve the algorithm's efficiency by introducing "pools" of variables (adjacent x_t values) which are updated simultaneously. We introduce temporary values $\boldsymbol{x}'$ and initialize them to the unconstrained least squares solution, $\boldsymbol{x}' = \boldsymbol{y}$. Initially all constraints are in the "passive set" and possible violations are fixed by subsequently adding the respective constraints to the "active set". Starting at $t = 2$ the algorithm moves to the right until a violation of the constraint $x'_\tau \geq x'_{\tau-1}$ at some time τ is encountered. Now the monotonicity constraint is added to the active set and enforced by setting $x'_\tau = x'_{\tau-1}$. (Supposing the opposite, i.e. $x'_\tau > x'_{\tau-1}$, we could move x'_τ and $x'_{\tau-1}$ by some small ϵ to decrease the objective without violating the constraints, yielding a proof by contradiction that the monotonicity constraint should be made "active" here—i.e., the constraint holds with strict equality.) We update the values $x'_\tau = x'_{\tau-1}$ at the two time steps to the best possible fit with constraints. Minimizing their contribution to the residual $(y_{\tau-1} - x'_{\tau-1})^2 + (y_\tau - x'_{\tau-1})^2$ by setting the derivative with respect to $x'_{\tau-1}$ to zero, $y_{\tau-1} - x'_{\tau-1} + y_\tau - x'_{\tau-1} = 0$, amounts to replacing the values with their average, $x'_{\tau-1} = x'_\tau = \frac{y_{\tau-1}+y_\tau}{2}$. However, this updated value can violate the constraint $x'_{\tau-1} \geq x'_{\tau-2}$ and we need to add this constraint to the active set and update $x'_{\tau-2}$ as well, $x'_{\tau-2} = x'_{\tau-1} = x'_\tau = \frac{y_{\tau-2}+y_{\tau-1}+y_\tau}{3}$, etc. In this manner the algorithm continues to back-average to the left as needed until we have backtracked to time t' where the constraint $x'_{t'} \geq x'_{t'-1}$ is already valid. Solving

$$\underset{x'_{t'}}{\text{minimize}} \quad \sum_{t=t'}^{\tau} (x'_{t'} - y_t)^2 \tag{6}$$

by setting the derivative to zero yields an update that corresponds to averaging

$$x'_{t'} = x'_{t'+1} = \ldots = x'_\tau = \frac{\sum_{t=t'}^{\tau} y_t}{\tau - t' + 1}. \tag{7}$$

The optimal solution that satisfies all constraints up to time τ has been found and the search

advances to the right again until detection of the next violation, backtracks again, etc. This process continues until the last value x_T is reached and having found the optimal solution we return $\boldsymbol{x} = \boldsymbol{x}'$.

Algorithm 1 Pool Adjacent Violators Algorithm (PAVA) for isotonic regression

Require: data $y_t \in \boldsymbol{y}$ at time of reading
initialize set of pools $\mathcal{P} \leftarrow \{\}$, data index $t \leftarrow 0$, pool index $i \leftarrow 0$
for y in $\boldsymbol{y}$ **do** ▷read next data point y
  $t \leftarrow t + 1$
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{(y, 1, t)\}$ ▷add pool
  while $i > 0$ and $v_{i+1} < v_i$ **do** ▷merge pools if necessary
    $\mathcal{P}_i \leftarrow \left(\frac{w_i v_i + w_{i+1} v_{i+1}}{w_i + w_{i+1}}, w_i + w_{i+1}, t_i\right)$
    remove $\mathcal{P}_{i+1}$
    $i \leftarrow i - 1$
  $i \leftarrow i + 1$
for (v, w, t) in $\mathcal{P}$ **do** ▷construct solution for all t
  for $\tau = 0, \ldots, w - 1$ **do** $x_{t+\tau} \leftarrow v$
return $\boldsymbol{x}$

In a worst case situation a constraint violation is encountered at every step of the forward sweep through the series. Updating all t values up to time t yields overall $\sum_{t=2}^{T} t = \frac{T(T+1)}{2} - 1$ updates and an $O(T^2)$ algorithm. However, note that when a violation is encountered the updated time points all share the same value (the average of the data at these time points, Eq 7) and it suffices to track this value just once for all these updated time points [30]. The constraints $x'_t \geq x'_{t-1}$ between the updated time points hold with equality $x'_t = x'_{t-1}$, and are part of the active set. In order to obtain a more efficient algorithm, cf. Algorithm 1 and S1 Video, we introduce "pools" or groups of the form (v_i, w_i, t_i) with value v_i, weight w_i and event time t_i where i indices the groups. Initially the ordered set of pools is empty. During the forward sweep through the data the next data point y_t is initialized as its own pool $(y_t, 1, t)$ and appended to the set of pools. Adjacent pools that violate the constraint $v_{i+1} \geq v_i$ are combined to a new pool $(\frac{w_i v_i + w_{i+1} v_{i+1}}{w_i + w_{i+1}}, w_i + w_{i+1}, t_i)$. Whenever pools i and $i + 1$ are merged, former pool $i + 1$ is removed. It is easy to prove by induction that these updates guarantee that the value of a pool is indeed the average of the corresponding data points (see S1 Appendix) without having to explicitly calculate it using Eq (7). The latter would be expensive for long pools, whereas merging two pools has $O(1)$ complexity independent of the pool lengths. With pooling the considered worst case situation results in a single pool and only its value and weight are updated at every step forward, yielding $O(T)$ complexity. Constructing the optimal solution x_t for all t in a final effort after the optimal pool partition has been reached is also $O(T)$. At convergence all constraints have been enforced; further note that convergence to the exact solution occurs after a finite number of steps, in contrast to interior point-methods which only approach the optimal solution asymptotically.

Online Active Set method to Infer Spikes (OASIS). Now we adapt the PAVA approach to problem (3). PAVA solves a regression problem subject to the constraint that the value at the current time bin must be greater than or equal to the last. The AR(1) model posits a more general but very similar constraint that bounds the rate of decay instead of enforcing monotonicity. The key insight is that problem (3) is a generalization of problem (5): if $p = 1$ in the AR model and we set $\gamma = 1$ (we skip the index of γ for a single AR coefficient) and $\lambda = 0$ in Eq (3) we obtain Eq (5). Therefore we focus first on the $p = 1$ case and deal with $p > 1$ and arbitrary calcium response kernels in the next section.

We begin by inserting the definition of $\hat{\boldsymbol{s}}$ (Eq 3). Using that $\hat{\boldsymbol{s}}$ is constrained to be non-negative yields for the sparsity penalty

$$\lambda \|\hat{\boldsymbol{s}}\|_1 = \lambda \mathbf{1}^\top \hat{\boldsymbol{s}} = \lambda \sum_{t=1}^{T} \sum_{k=1}^{T} G_{k,t} \hat{c}_t = \lambda \sum_{t=1}^{T} (1 - \gamma + \gamma \delta_{tT}) \hat{c}_t = \sum_{t=1}^{T} \mu_t \hat{c}_t = \boldsymbol{\mu}^\top \hat{\boldsymbol{c}} \tag{8}$$

with $\mu_t \coloneqq \lambda(1 - \gamma + \gamma\delta_{tT})$ (with δ denoting Kronecker's delta) by noting that the sum of the last column of G is 1, whereas all other columns sum to $(1 - \gamma)$.

Now the problem

$$\underset{\hat{c}}{\text{minimize}} \quad \frac{1}{2} \sum_{t=1}^{T} (\hat{c}_t - y_t)^2 + \sum_{t=1}^{T} \mu_t \hat{c}_t \quad \text{subject to} \quad \hat{c}_{t+1} \geq \gamma \hat{c}_t \geq 0 \quad \forall t \tag{9}$$

shares some similarity to isotonic regression with the constraint $\hat{c}_{t+1} \geq \hat{c}_t$ (Eq 5). However, our constraint $\hat{c}_{t+1} \geq \gamma \hat{c}_t$ bounds the rate of decay instead of enforcing monotonicity. Thus we need to generalize PAVA to handle the additional factor γ.

For clarity we mimic our approach from the last section: we first present the algorithm in a way that conveys its core ideas, and then improve the algorithm's efficiency using pools. We introduce temporary values $\boldsymbol{c}'$ and initialize them to the unconstrained least squares solution, $\boldsymbol{c}' = \boldsymbol{y} - \boldsymbol{\mu}$. Starting at $t = 2$ one moves forward until a violation of the constraint $c'_\tau \geq \gamma c'_{\tau-1}$ at some time τ is detected (Fig 2A). Updating the two time steps by minimizing $\frac{1}{2}\left(y_{\tau-1} - c'_{\tau-1}\right)^2 + \frac{1}{2}\left(y_\tau - \gamma c'_{\tau-1}\right)^2 + \mu_{\tau-1} c'_{\tau-1} + \mu_\tau \gamma c'_{\tau-1}$ yields an updated value $c'_{\tau-1}$. However, this updated value can violate the constraint $c'_{\tau-1} \geq \gamma c'_{\tau-2}$ and we need to update $c'_{\tau-2}$ as well, etc., until we have backtracked some Δt steps to time $t' = \tau - \Delta t$ where the constraint $c'_{t'} \geq \gamma c'_{t'-1}$ is already valid. At most one needs to backtrack to the most recent spike, because $c'_{t'} > \gamma c'_{t'-1}$ at spike times t' (Eq 1). Solving

$$\underset{c'_{t'}}{\text{minimize}} \quad \frac{1}{2} \sum_{t=0}^{\Delta t} (\gamma^t c'_{t'} - y_{t+t'})^2 + \sum_{t=0}^{\Delta t} \mu_{t+t'} \gamma^t c'_{t'} \tag{10}$$

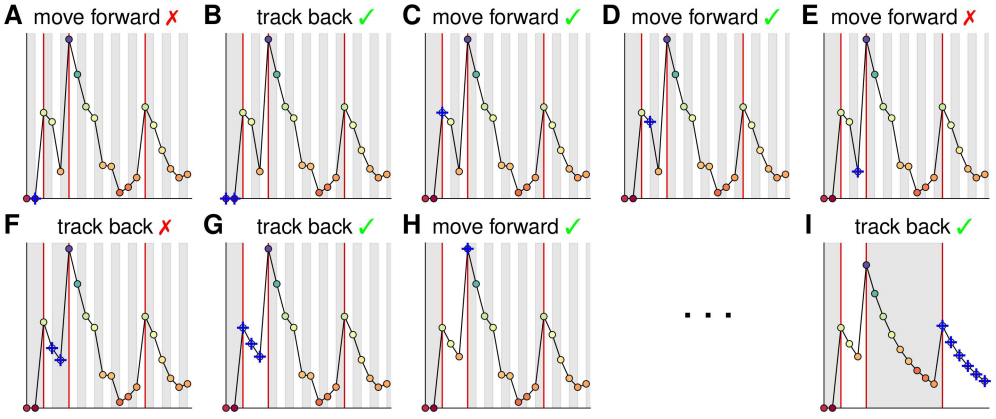

Fig 2. Illustration of OASIS for an AR(1) process (see S2 Video). Red lines depict true spike times. The shaded background shows how the time points are gathered in pools. The pool currently under consideration is indicated by the blue crosses. A constraint violation is encountered for the second time step **(A)** leading to backtracking and merging **(B)**. The algorithm proceeds moving forward **(C-E)** until the next violation occurs **(E)** and triggers backtracking and merging **(F-G)** as long as constraints are violated. When the most recent spike time has been reached **(G)** the algorithm proceeds forward again **(H)**. The process continues until the end of the series has been reached **(I)**. The solution is obtained and pools span the inter-spike-intervals.

https://doi.org/10.1371/journal.pcbi.1005423.g002

by setting the derivative to zero yields

$$c'_{t'} = \frac{\sum_{t=0}^{\Delta t}(y_{t+t'} - \mu_{t+t'})\gamma^t}{\sum_{t=0}^{\Delta t}\gamma^{2t}} \tag{11}$$

and the next values are updated according to $c'_{t'+t} = \gamma^t c'_{t'}$ for $t = 1, \ldots, \Delta t$. Note the similarity of Eqs (7) and (11), which differs by weighting the summands by powers of γ due to the altered constraints, and by subtracting $\boldsymbol{\mu}$ from the data $\boldsymbol{y}$ due to the sparsity penalty. (Along the way it is worth noting that, because a spike induces a calcium response described by kernel $\boldsymbol{h}$ with components $h_{1+t} = \gamma^t$, $c'_{t'}$ could be expressed in the more familiar regression form as $\frac{\boldsymbol{h}^\top_{1:\Delta t+1}(\boldsymbol{y}-\boldsymbol{\mu})_{t':\tau}}{\boldsymbol{h}^\top_{1:\Delta t+1}\boldsymbol{h}_{1:\Delta t+1}}$, where we used the notation $\boldsymbol{v}_{i:j}$ to describe a vector formed by components i to j of $\boldsymbol{v}$.) Now one moves forward again (Fig 2C–2E) until detection of the next violation (Fig 2E), backtracks again to the most recent spike (Fig 2G), etc. Once the end of the time series is reached (Fig 2I) we have found the optimal solution and set $\hat{\boldsymbol{c}} = \boldsymbol{c}'$.

While this yields a valid algorithm, it frequently updates each value c'_t and recalculates the full sums in Eq (11) for each step of backtracking. A similar algorithm has been suggested by [31] for the problem without sparsity penalty. However, it passes through the time series in reverse direction, from its end to its beginning, and is thus not applicable to online processing. It considers directly the deconvolved activity $\hat{\boldsymbol{s}}$ and efficiently does not update all time steps but only suspected spike times. However, their algorithm uses the inefficient updates of Eq 11, rendering it an $O(T^2)$ algorithm.

As in PAVA, next we introduce "pools" into the algorithm; these are of critical importance in order to obtain a true $O(T)$ algorithm. In PAVA these pools serve as sufficient statistics summarizing the data between jumps in the estimated output x_t; here the pools summarize the data between estimated spike times, where the estimated calcium signal $\hat{c}_t$ jumps. Pools are now tuples of the form (v_i, w_i, t_i, l_i) with value v_i, weight w_i, event time t_i and pool length l_i. Here we explicitly track the pool length, which was identical to its weight for PAVA. Initially the ordered set of pools is empty. During the forward sweep through the data the next data point y_t is initialized as its own pool $(y_t - \mu_t, 1, t, 1)$ and appended to the set of pools. During backtracking pools get combined and only the first value $v_i = c'_{t_i}$ is explicitly considered, while the other values are merely defined implicitly via $c'_{t+1} = \gamma c'_t$. The constraint $c'_{t+1} \geq \gamma c'_t$ translates to $v_{i+1} \geq \gamma^{l_i} v_i$ as the criterion determining whether pools need to be combined. The introduced weights allow efficient value updates whenever pools are merged by avoiding recalculating the sums in Eq (11). Values are updated according to

$$v_i \leftarrow \frac{w_i v_i + \gamma^{l_i} w_{i+1} v_{i+1}}{w_i + \gamma^{2l_i} w_{i+1}} \tag{12}$$

where the denominator is the new weight of the pool and the pool lengths are summed

$$w_i \leftarrow w_i + \gamma^{2l_i} w_{i+1} \tag{13}$$

$$l_i \leftarrow l_i + l_{i+1} \tag{14}$$

Whenever pools i and $i + 1$ are merged, former pool $i + 1$ is removed. It is easy to prove by induction that the updates according to Eqs (12–14) guarantee that Eq (11) holds for all values (see S1 Appendix).

Analogous to PAVA, the updates solve Eq (9) not just greedily but optimally, finding the exact solution to the convex problem in $O(T)$. One important point (especially relevant for online use) is that the computation time per observation timestep is not fixed but random, since

we might have to backtrack to update an unpredictable number of pools. We found empirically that, over a wide range of hyperparameters, in about 80% of the cases 0–1 merge operation was performed per observation timestep. With less than 0.5% probability four or more merges were necessary; in all our experiments so far, never more than seven merges were needed.

The final algorithm is summarized in Algorithm 2 and illustrated in Fig 2 as well as in S2 Video. Comparing Algorithm 1 with 2 clearly reveals the modifications made and shows that for $\gamma = 1$ and $\lambda = 0$ the algorithm reduces to PAVA.

Algorithm 2 Fast online deconvolution algorithm for AR1 processes with positive jumps

Require: decay factor γ, regularization parameter λ, data $y_t \in \boldsymbol{y}$ at time of reading
initialize set of pools $\mathcal{P} \leftarrow \{\}$, time index $t \leftarrow 0$, pool index $i \leftarrow 0$, solution $\hat{\boldsymbol{s}} \leftarrow 0$
for y in $\boldsymbol{y}$ **do** ▷read next data point y
  $t \leftarrow t + 1$
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{(y - \lambda(1 - \gamma + \gamma\delta_{tT}), 1, t, 1)\}$ ▷add pool
  while $i > 0$ and $v_{i+1} < \gamma^{l_i} v_i$ **do** ▷merge pools if necessary
    $\mathcal{P}_i \leftarrow \left(\frac{w_i v_i + \gamma^{l_i} w_{i+1} v_{i+1}}{w_i + \gamma^{2l_i} w_{i+1}}, w_i + \gamma^{2l_i} w_{i+1}, t_i, l_i + l_{i+1}\right)$ ▷Eqs (12–14)
    remove $\mathcal{P}_{i+1}$
    $i \leftarrow i - 1$
  $i \leftarrow i + 1$
for (v, w, t, l) in $\mathcal{P}$ **do** ▷construct solution for all t[†]
  for $\tau = 0, \ldots, l - 1$ **do** $\hat{c}_{t+\tau} \leftarrow \gamma^\tau \max(0, v)$ ▷enforce $\hat{c}_t \geq 0$ via max
  if $t > 1$ **then** $\hat{s}_t \leftarrow \hat{c}_t - \gamma \hat{c}_{t-1}$
return $\hat{\boldsymbol{c}}, \hat{\boldsymbol{s}}$

[†]For online estimates of $\hat{\boldsymbol{c}}$ and $\hat{\boldsymbol{s}}$ the solution can be constructed within the loop over $\boldsymbol{y}$ not just after it.

Dual formulation with hard noise constraint. The formulation above contains a troublesome free sparsity parameter λ (implicit in $\boldsymbol{\mu}$). A more robust deconvolution approach chooses the sparsity implicitly by inclusion of the residual sum of squares (RSS) as a hard constraint and not as a penalty term in the objective function [13]. The expected RSS satisfies $\langle \|\boldsymbol{c} - \boldsymbol{y}\|^2 \rangle = \sigma^2 T$ and by the law of large numbers $\|\boldsymbol{c} - \boldsymbol{y}\|^2 \approx \sigma^2 T$ with high probability, leading to the constrained problem

$$\underset{\hat{\boldsymbol{c}}, \hat{\boldsymbol{s}}}{\text{minimize}} \quad \|\hat{\boldsymbol{s}}\|_1 \quad \text{subject to} \quad \hat{\boldsymbol{s}} = G\hat{\boldsymbol{c}} \geq 0 \quad \text{and} \quad \|\hat{\boldsymbol{c}} - \boldsymbol{y}\|^2 \leq \hat{\sigma}^2 T. \tag{15}$$

(As noted above, we estimate σ using the power spectral estimator described in [13]; see also [8] for a similar approach.)

We will solve this problem by increasing λ in the dual formulation until the noise constraint is tight. We start with some small λ, e.g. $\lambda = 0$, to obtain a first partitioning into pools $\mathcal{P}$, cf. Fig 3A. From Eqs (11–13) along with the definition of $\boldsymbol{\mu}$ (Eq 8) it follows that given the solution (v_i, w_i, t_i, l_i), where

$$v_i = \frac{\sum_{t=0}^{l_i-1} (y_{t_i+t} - \mu_{t_i+t}) \gamma^t}{\sum_{t=0}^{l_i-1} \gamma^{2t}} = \frac{\sum_{t=0}^{l_i-1} (y_{t_i+t} - \lambda(1 - \gamma + \gamma\delta_{t_i+t,T})) \gamma^t}{w_i} \tag{16}$$

for some λ, the solution (v_i', w_i', t_i', l_i') for $\lambda + \Delta\lambda$ is

$$v_i' = v_i - \Delta\lambda \frac{\sum_{t=0}^{l_i-1} (1 - \gamma + \gamma\delta_{t_i+t,T}) \gamma^t}{w_i} = v_i - \Delta\lambda \frac{1 - \gamma^{l_i}(1 - \delta_{iz})}{w_i} \tag{17}$$

where z is the index of the last pool and because pools are updated independently we make the

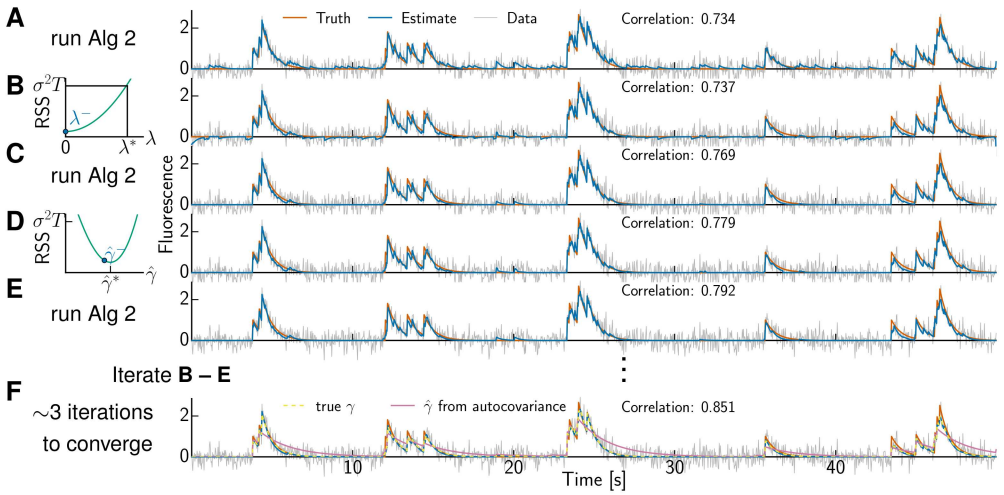

Fig 3. Optimizing sparsity parameter λ and AR coefficient $\hat{\gamma}$. (A) Running the active set method, with conservatively small estimate $\hat{\gamma}$, yields an initial *denoised* estimate (blue) of the data (gray) roughly capturing the truth (red). We also report the correlation between the *deconvolved* estimate and true spike train as a direct measure for the accuracy of spike train inference. **(B)** Updating sparsity parameter λ according to Eq (18) such that RSS = $\sigma^2 T$ (left) shifts the current estimate downward (right, blue). **(C)** Running the active set method enforces the constraints again and is fast due to warm-starting. **(D)** Updating $\hat{\gamma}$ by minimizing the polynomial function RSS($\hat{\gamma}$) and **(E)** running the warm-started active set method completes one iteration, which yields already a decent fit. **(F)** A few more iterations improve the solution further. The obtained estimate (blue) is hardly distinguishable from the one obtained with known true γ (yellow dashed trace, plotted in addition to the traces in A-E, is on top of blue solid line). Note that determining $\hat{\gamma}$ based on the autocovariance (additionally plotted purple trace) yields a crude solution that even misses spikes (at 24.6 s and 46.5 s).

https://doi.org/10.1371/journal.pcbi.1005423.g003

approximation that no changes in the pool structure occur. Inserting Eq (17) into the noise constraint (Eq 15) results in

$$\sum_{i=1}^{z}\sum_{t=0}^{l_i-1}\left(\left(v_i-\Delta\lambda\frac{1-\gamma^{l_i}(1-\delta_{iz})}{w_i}\right)\gamma^t-y_{t_i+t}\right)^2=\hat{\sigma}^2T \tag{18}$$

and solving the quadratic equation for Δλ yields

$$\Delta\lambda=\frac{-\beta+\sqrt{\beta^2-4\alpha\epsilon}}{2\alpha} \tag{19}$$

with $\alpha=\sum_{i,t}\xi_{it}^2$, $\beta=2\sum_{i,t}\chi_{it}\xi_{it}$ and $\epsilon=\sum_{i,t}\chi_{it}^2-\hat{\sigma}^2T$ where $\xi_{it}=\frac{1-\gamma^{l_i}(1-\delta_{iz})}{w_i}\gamma^t$ and $\chi_{it}=y_{t_i+t}-v_i\gamma^t$.

The solution Δλ provides a good approximate proposal step for updating the pool values v_i (using Eq 17). Since this update proposal is only approximate it can give rise to violated constraints (e.g., negative values of v_i). To satisfy all constraints Algorithm 2 is run to update the pool structure, cf. Fig 3C, but with a *warm start*: we initialize with the current set of merely z pools $\mathcal{P}'$ instead of the T pools for a cold start (Alg 2, line 4). This step returns a set of v_i values that satisfy the constraints and may merge pools (i.e., delete spikes); then the procedure (update λ then rerun the warm-started Algorithm 2) can be iterated until no further pools need to be merged, at which point the procedure has converged. In practice this leads to an increasing sequence of λ values (corresponding to an increasingly sparse set of spikes), and no pool-split (i.e., add-spike) moves are necessary. (Note that it is possible to cheaply detect any

violations of the KKT conditions in a candidate solution; if such a violation is detected, the corresponding pool could be split and the warm-started Algorithm 2 run locally near the detected violations. However, as we noted, due to the increasing λ sequence we did not find this step to be necessary in the examples examined here.)

This warm-starting approach brings major speed benefits: after the residual is updated following a λ update, the computational cost of the algorithm is linear in the number of pools z, hence warm starting drastically reduces computational costs from $k_1 T$ to $k_2 z$ with proportionality constants k_1 and k_2: if no pool boundary updates are needed then after warm starting the algorithm only needs to pass once through all pools to verify that no constraint is violated, whereas a cold start might involve a couple passes over the data to update pools, so k_2 is typically significantly smaller than k_1, and z is typically much smaller than T (especially in sparsely-spiking regimes).

Additional baseline. For ease of exposition we thus far assumed no offsetting baseline. Adding a known baseline $b \neq 0$ the problem reads

$$\underset{\hat{\boldsymbol{c}},\hat{\boldsymbol{s}}}{\text{minimize}} \quad \tfrac{1}{2}\,\|\, b\mathbf{1} + \hat{\boldsymbol{c}} - \boldsymbol{y}\,\|^2 + \lambda \|\, \hat{\boldsymbol{s}}\,\|_1 \quad \text{subject to} \quad \hat{\boldsymbol{s}} = G\hat{\boldsymbol{c}} \geq 0. \tag{20}$$

For known baseline one merely needs to initialize OASIS by subtracting not only the sparsity parameter $\boldsymbol{\mu}(\lambda)$ from the data $\boldsymbol{y}$, cf. Eq (11) and Algorithm 2, but also the baseline b. The fluorescence $\hat{\boldsymbol{c}}$ depends only on the sum $\boldsymbol{\phi} = b\mathbf{1} + \boldsymbol{\mu}$.

If the baseline is not known, we want to optimize it too by solving the noise constrained dual problem

$$\underset{\hat{b},\hat{\boldsymbol{c}},\hat{\boldsymbol{s}}}{\text{minimize}} \quad \|\hat{\boldsymbol{s}}\|_1 \quad \text{subject to} \quad \hat{\boldsymbol{s}} = G\hat{\boldsymbol{c}} \geq 0 \quad \text{and} \quad \|\, \hat{b}\mathbf{1} + \hat{\boldsymbol{c}} - \boldsymbol{y}\,\|^2 \leq \hat{\sigma}^2 T. \tag{21}$$

We denote all except the differing last component of $\boldsymbol{\mu}$ by $\mu = \lambda(1-\gamma)$ (Eq 8) and of $\boldsymbol{\phi}$ by $\phi = b + \lambda(1-\gamma)$. ϕ is the total shift applied to the data (except for the last time step) due to the baseline and sparsity penalty before running OASIS. We increase ϕ until the noise constraint is tight. ϕ can be initialized by $\min y_t$ or better by a small percentile of $\boldsymbol{y}$, e.g. 15%. Once OASIS has been run with some ϕ the baseline $\hat{b}$ is obtained by minimizing the objective Eq (20) with respect to it, yielding $\hat{b} = \langle \boldsymbol{y} - \hat{\boldsymbol{c}} \rangle = \frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{c}_t)$, and the sparsity parameter is $\mu = \phi - \hat{b}$. Appropriately adding $\hat{b}$ to Eq (18)

$$\sum_{i=1}^{z}\sum_{t=0}^{l_i-1}\left(\left(v_i - \Delta\phi\,\frac{1-\gamma^{l_i}(1-\delta_{iz})}{(1-\gamma)w_i}\right)\gamma^t - y_{t_i+t} + \hat{b}\right)^2 = \hat{\sigma}^2 T \tag{22}$$

and plugging the analytic expression $\hat{b} = \frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{c}_t) =$

$\frac{1}{T}\sum_{j=1}^{z}\sum_{\tau=0}^{l_j-1}\left(y_{t_j+\tau} - \left(v_j - \Delta\phi\,\frac{1-\gamma^{l_j}(1-\delta_{jz})}{(1-\gamma)w_j}\right)\gamma^\tau\right)$ into Eq (22) to account for the changing baseline, we obtain an estimate of $\Delta\phi$ using a block coordinate update of ϕ and $\hat{b}$. Solving the ensuing quadratic equation for $\Delta\phi$, yields

$$\Delta\phi = \frac{-\beta + \sqrt{\beta^2 - 4\alpha\epsilon}}{2\alpha} \tag{23}$$

with $\alpha = \sum_{i,t}\xi_{it}^2$, $\beta = 2\sum_{i,t}\chi_{it}\,\xi_{it}$ and $\epsilon = \sum_{i,t}\chi_{it}^2 - \hat{\sigma}^2 T$ where $\xi_{it} = \frac{1-\gamma^{l_i}(1-\delta_{iz})}{(1-\gamma)w_i}\gamma^t - \sum_j \frac{(1-\gamma^{l_j}(1-\delta_{jz}))^2}{T(1-\gamma)^2 w_j}$ and $\chi_{it} = y_{t_i+t} - v_i\gamma^t - \frac{1}{T}\sum_{j,\tau}(y_{t_j+\tau} - v_j\gamma^\tau)$. All pools are updated according to

$v_i' = v_i - \Delta\phi\frac{1-\gamma^{l_i}(1-\delta_{iz})}{(1-\gamma)w_i}$, cf. Eq (17). To satisfy all constraints Algorithm 2 is run, warm-started by initializing with the current set of pools.

Optimizing the AR coefficient. Thus far the parameter γ has been known or been estimated based on the autocovariance function. We can improve upon this estimate by optimizing $\hat{\gamma}$ as well, which is illustrated in Fig 3. After updating λ (and $\hat{b}$) followed by running Algorithm 2, we perform a coordinate descent step in $\hat{\gamma}$ that minimizes the RSS, cf. Fig 3D. The RSS as a function of $\hat{\gamma}$ is a high order polynomial, cf. Eq (11), and we need to settle for numerical solutions of

$$\hat{\gamma} = \arg\min_{\gamma} \quad \sum_{i=1}^{z}\sum_{t=0}^{l_i-1}\left(\hat{b} + \frac{\sum_{\tau=0}^{l_i-1}(y_{t_i+\tau} - \mu_{t_i+\tau})\gamma^{\tau}}{\sum_{\tau=0}^{l_i-1}\gamma^{2\tau}}\gamma^{t} - y_{t_i+t}\right)^2. \tag{24}$$

We used Brent's method [32] with bounds $0 \leq \hat{\gamma} < 1$ to solve this problem. One iteration consists now of steps B-E in Fig 3, while for known γ only B-C were necessary. If optimizing the baseline too, we obtained better results by minimizing the RSS jointly with respect to $\hat{\gamma}$ and $\hat{b}$ using L-BFGS-B [33] instead of keeping the baseline $\hat{b}$ fixed.

Faster optimization of hyperparameters. We have presented methods to estimate the hyperparameters λ, b and γ, which require a handful of warm-started iterations of OASIS. To gain further speed-ups these parameters can be estimated on decimated data. When downsampling by a factor k, the average of k subsequent frames is calculated, the noise $\hat{\sigma}$ divided by a factor $\sqrt{k}$ and the initial estimate of the AR coefficient scaled to $\hat{\gamma}^k$. Alternatively, one could estimate σ and γ based on the decimated data. Once the hyperparameters have been obtained, the corresponding inverse transformations are performed: $\hat{\gamma} \to \hat{\gamma}^{\frac{1}{k}}$, $\hat{b} \to \hat{b}$ and $\lambda \to \lambda\frac{1-\hat{\gamma}}{1-\hat{\gamma}^{1/k}}$ such that the shrinkage $\mu = \lambda(1-\hat{\gamma})$ due to the penalty term stays invariant. The final run of OASIS on the full data is warm started using the solution obtained on the decimated data. Data points that are not in the proximity of a spike of the downsampled solution are already combined into large pools, instead of initializing each data point as its own separate pool. More precisely, if the deconvolved decimated data has positive values at times $\{t_i\}$, for deconvolving the full data time steps $\bigcup_i\{(k-1)t_i, \ldots, (k+1.5)t_i\}$ are initialized as individual pools, while the remaining time steps are pooled together into bigger pools, separated from each other by the individual ones, with values given by Eq (11) and weights by its denominator.

In particular the estimation of the AR coefficient γ is computationally burdensome, because it involves expensive repeated evaluations of the RSS in order to minimize it as function of $\hat{\gamma}$ (and $\hat{b}$). The computing time depends linearly on the number of pools z and we gain further speed-ups by restricting the attention to merely a subset of pools. In particular, because γ can be well estimated based on large isolated calcium events, we restrict the calculation of the RSS to the pools with largest product of value and length. A large value indicates a large event and a long pool an isolated event. We present detailed results in the Results section, indicating that altogether we can save about an order of magnitude computation with the greatest savings obtained by reducing the optimization of $\hat{\gamma}$ from $O(z)$ to $O(1)$.

It is also worth noting that the hyperparameter estimation discussed above is performed in 'batch' mode, not online. However, once good hyperparameter values are obtained on a short initial batch we can switch into online mode (with the hyperparameters held fixed) and handle the remaining data in a stream.

Hard shrinkage and ℓ_0 penalty. It is well-known that ℓ_1 penalization results in "soft-thresholding" [34], in which small values are zeroed out and large values are shifted to lower values (where the size of this shift is proportional to the penalty λ). We can perform hard

instead of soft thresholding (avoiding this shrinkage of large values) by replacing the sparsity penalty by a constraint on the minimum spike size $s_{\min}$. The problem

$$\underset{\hat{\boldsymbol{c}},\hat{\boldsymbol{s}}}{\text{minimize}} \quad \frac{1}{2}\|\hat{\boldsymbol{c}} - \boldsymbol{y}\|^2 \quad \text{subject to} \quad \hat{\boldsymbol{s}} = G\hat{\boldsymbol{c}} \text{ with } \hat{s}_t \geq s_{\min} \text{ or } \hat{s}_t = 0 \tag{25}$$

is non-convex and we are not guaranteed to find the global minimum. However, we obtain a good local minimum by merely changing the condition to merge pools from $v_{i+1} < \gamma^{l_i} v_i$ to $v_{i+1} < \gamma^{l_i} v_i + s_{\min}$, modifying lines 3 and 5 in Algorithm 2.

Now we must choose a value for $s_{\min}$. In many cases we found that simply setting $s_{\min}$ as a small multiple of the noise level led to good results. If the scaling factor a (Eq 2) relating fluorescence to action potentials was known, we could properly normalize the spike train such that $\hat{s}_t = 1$ corresponds to one spike and choose $s_{\min} = 0.5$, or a slightly higher value to avoid splitting one spike into two of size 0.5. However, often the factor is unknown or difficult to estimate, rendering the choice of $s_{\min}$ cumbersome. Analogous to the variation of λ, we can start with $s_{\min} = 0$ and increase it until the RSS crosses the $\sigma^2 T$ threshold by sequentially removing the smallest 'spike' and merging the pools it used to separate. By maximizing $s_{\min}$ under the noise constraint we minimize the number of non-zero values of $\hat{\boldsymbol{s}}$. De facto, we try to find a parsimonious description of the data by minimizing the number of non-zero values of $\hat{\boldsymbol{s}}$, thus solving a sparsity problem with ℓ_0 penalty:

$$\underset{\hat{\boldsymbol{c}},\hat{\boldsymbol{s}}}{\text{minimize}} \quad \|\hat{\boldsymbol{s}}\|_0 \quad \text{subject to} \quad \hat{\boldsymbol{s}} = G\hat{\boldsymbol{c}} \geq 0 \quad \text{and} \quad \|\hat{\boldsymbol{c}} - \boldsymbol{y}\|^2 \leq \hat{\sigma}^2 T \tag{26}$$

Instead of sequentially removing the smallest 'spike' we actually obtained the best performance by sequentially adding spikes at the highest values of the ℓ_1-solution $\hat{\boldsymbol{s}}$ until the RSS is smaller than $\hat{\sigma}^2 T$. While the updates resemble those of matching pursuit [35], in practice we found that adding spikes at the positions suggested by the ℓ_1-solution yields better results than matching pursuit (which adds spikes at positions that greedily lead to the highest RSS reduction per step). Specifically, we found that often matching pursuit cannot resolve spikes in close proximity, but instead results in erroneous placement of one big spike as an explanation for all nearby spikes. Instead of merging pools we now need to split pools. Denoting the time where to add a spike by t_s, i.e. the time where the ℓ_1-solution has its highest value after ruling out times where spikes have already been added, one searches for the pool i in which it falls, i.e. $t_i < t_s < t_i + l_i$. Pool i gets updated as $l'_i = t_s - t_i$, $w'_i = \sum_{t=0}^{l'_i-1} \gamma^{2t}$, and $v'_i = \sum_{t=0}^{l'_i-1} y_{t+t_i}\gamma^t / w'_i$, which follows directly from Eq (11) with $\mu_t = 0$. All pool indices greater than i are increased by one and a new pool is inserted after pool i with $l'_{i+1} = l_i - l'_i$, $t'_{i+1} = t_s$, $w'_{i+1} = \sum_{t=0}^{l'_{i+1}-1} \gamma^{2t}$, and $v'_{i+1} = \sum_{t=0}^{l'_{i+1}-1} y_{t+t_s}\gamma^t / w'_{i+1}$.

As is the case with all optimized hyperparameters, once we have obtained a decent estimate of $s_{\min}$ on an initial subset of the data we can switch back into online mode. In online mode our algorithm is typically faster than matching pursuit, since matching pursuit requires updating $O(\Delta)$ points of the residual with each update, where Δ is the length of the calcium transient (in number of frames).

Generalization beyond the AR(1) case

A greedy solution for the AR(p>1) processes. An AR(1) process models the calcium response to a spike as an instantaneous increase followed by an exponential decay. This is a good description when the fluorescence rise time constant is small compared to the length of a time-bin, e.g. when using GCaMP6f [36] with a slow imaging rate. For fast imaging rates and slow indicators such as GCaMP6s it is more accurate to explicitly model the finite rise time.

Typically we choose an AR(2) process, though more structured responses (e.g. multiple decay time constants) can also be modeled with higher values for the order p.

For an AR(p) process the sparsity penalty $\lambda\|\hat{\boldsymbol{s}}\|_1$ can again be expressed as $\boldsymbol{\mu}^\top\hat{\boldsymbol{c}}$, because

$$\lambda\,\|\hat{\boldsymbol{s}}\|_1 = \lambda\mathbf{1}^\top\hat{\boldsymbol{s}} = \lambda\sum_{t=1}^{T}\sum_{k=1}^{T} G_{k,t}\hat{c}_t = \lambda\sum_{t=1}^{T}(1-\sum_{i=1}^{\min(p,T-t)}\gamma_i)\hat{c}_t = \sum_{t=1}^{T}\mu_t\hat{c}_t = \boldsymbol{\mu}^\top\hat{\boldsymbol{c}}, \tag{27}$$

with $\mu_t := \lambda(1-\sum_{i=1}^{\min(p,T-t)}\gamma_i)$, by evaluating the column sums of G. For $p > 1$ the dynamics are no longer first-order Markov and the next value depends not only on the current but on possibly multiple previous time steps. Now following along the lines of the previous section just leads to a greedy, approximate solution; we will present an exact algorithm later. We use matrix- and vector notation to describe the dynamics of c_t. Let the transition matrix A, multi time step calcium vectors $\boldsymbol{\zeta}_t$, and vector $\boldsymbol{e}$ be defined as

$$A = \begin{pmatrix} \gamma_1 & \gamma_2 & \cdots & \gamma_p \\ 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \end{pmatrix} \qquad \boldsymbol{\zeta}_t = \begin{pmatrix} c_t \\ c_{t-1} \\ \vdots \\ c_{t-p+1} \end{pmatrix} \qquad \boldsymbol{e} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{28}$$

The calcium dynamics is given by $\boldsymbol{\zeta}_t = A\boldsymbol{\zeta}_{t-1} + s_t\boldsymbol{e}$. Analogously to the AR(1) case we derive an algorithm that moves through the time series until it finds a violation of the constraint $c'_\tau \geq \boldsymbol{e}^\top A\boldsymbol{\zeta}'_{\tau-1}$ for some time τ, updates c'_τ and $\boldsymbol{\zeta}'_{\tau-1}$, and backtracks further until the updates do not violate any constraints at previous time steps. Note that we also implicitly have constraints on $\boldsymbol{\zeta}_t$, enforcing the fact that $\boldsymbol{\zeta}_{t+1}$ is mostly a time-shifted version of $\boldsymbol{\zeta}_t$.

Assuming we need to backtrack by Δt steps and introducing again $t' = \tau - \Delta t$, the objective is to minimize $\sum_{t=t'}^{\tau}\left(\frac{1}{2}\left(c'_t - y_t\right)^2 + \mu_t c'_t\right)$ with respect to $c'_{t'}$ under the active constraints $\boldsymbol{\zeta}_t = A\boldsymbol{\zeta}_{t-1}$ for $t = t'+1, \ldots, \tau$. Plugging in the constraints on the dynamics the objective reads

$$\underset{c'_{t'}}{\text{minimize}} \quad \frac{1}{2}\sum_{t=0}^{\Delta t}\left(\boldsymbol{e}^\top A^t\boldsymbol{\zeta}'_{t'} - y_{t+t'}\right)^2 + \sum_{t=0}^{\Delta t}\mu_{t+t'}\boldsymbol{e}^\top A^t\boldsymbol{\zeta}'_{t'}. \tag{29}$$

Setting the derivative with respect to $c'_{t'}$ to zero and solving for $c'_{t'}$ yields

$$c'_{t'} = \frac{\sum_{t=0}^{\Delta t}\left(y_{t+t'} - \mu_{t+t'} - \sum_{k=2}^{p}\left(A^t\right)_{1,k}c'_{t'+1-k}\right)\left(A^t\right)_{1,1}}{\sum_{t=0}^{\Delta t}\left(A^t\right)_{1,1}^2} \tag{30}$$

where $\left(A^t\right)_{1,1}^2$ denotes the square of the entry in the first row and column in the matrix obtained as t-th matrix power of A. Again, note that these entries describe the calcium kernel $\boldsymbol{h}$ with components $h_{1+t} = \left(A^t\right)_{1,1}$. Eq (30) reduces to Eq (11) for $p = 1$ where A is just a 1×1-matrix with entry γ. The next values are updated according to $c'_{t'+t} = \sum_{k=1}^{p}\gamma_k c'_{t'+t-k}$ for $t = 1, \ldots, \Delta t$.

We derive again an efficient formulation of the algorithm using pools. Considering the denominator in Eq (30) as a weight in analogy to the AR(1) case and calculating the weighted sum upon merging of pools is not valid for $p > 1$ because in general $\left(A^t\right)_{1,1}\left(A^u\right)_{1,1} \neq \left(A^{t+u}\right)_{1,1}$. Introducing pools is still useful as it allows us to keep track of only a small number of p elements in each pool. While for the case of AR(1) we only kept track of each group's first element, we now keep track of the first as well as the $p-1$ last elements. In order to speed up the update in Eq (30), we can precompute the powers of A and store $\left(A^t\right)_{1,:}$ in memory.

Note that only the powers up to the maximal inter-spike-interval are needed, which can be much smaller than T; of course, for very large values of t, $(A^t)_{1,:} \approx 0$, by the stability of A; thus for high powers the entries of $(A^t)_{1,:}$ can also be well approximated by a quickly computable exponential function or simply be truncated. Analogous to the case $p = 1$, we can also impose a constraint on the minimum spike size $s_{\min}$ at the expense of having to deal with a non-convex problem by merely changing the condition to merge pools from $v_{i+1} < (A^{l_i})_{1,1} v_i + (A^{l_i})_{1,2} u_{i-1}$ to $v_{i+1} < (A^{l_i})_{1,1} v_i + (A^{l_i})_{1,2} u_{i-1} + s_{\min}$ where v_i and u_i denote the first and last value of pool i.

According to Eq (30) the solution is a linear function of $\boldsymbol{\mu}$, and hence of λ. Thus the hard noise constraint for the RSS $\|\boldsymbol{c} - \boldsymbol{y}\|^2 = \sigma^2 T$ is a quadratic equation in λ, that can be solved analytically, under the assumption of invariant pool structure analogous to above case of AR(1), but involving more lengthy expressions which we state explicitly in S1 Appendix. Updating all pools independently according to Eq (30) can give rise to violated constraints, requiring us to rerun the algorithm, warm-started by initializing with the current set of pools, as described above. After 2–3 iterations no pools need to be merged and the final solution has been found. We can again interleave an update step for optimizing the parameters γ_i, as described above.

Online Non-Negative Least Squares (ONNLS). We noted above that Eq (30) is not first-order Markovian: it includes a dependency on $p - 1$ previous time steps and hence in general the previous pool. In updating only the first value within a pool and using the current values of the $p - 1$ last values of the previous pool within the update Eq (30), we actually performed greedy updates. These greedy updates can yield remarkably good results, in particular for long pools, such that the last value is already well constrained by a number of data points and hardly affected by the next pool. Nonetheless, in some cases these greedy updates lead to errors in the timing of inferred activity, in particular when the rise time of the calcium response is slow compared to the frame rate. The method described in this section can be used to correct these small errors. It is again an active set method that can be run in online mode; however, the method introduced above is a *dual* active set method, whereas here we describe a *primal* active set method.

We begin by reformulating the problem as

$$\underset{\hat{\boldsymbol{s}}}{\text{minimize}} \quad \frac{1}{2} \| K\hat{\boldsymbol{s}} - \boldsymbol{y} \|^2 + \lambda \| \hat{\boldsymbol{s}} \|_1 \quad \text{subject to} \quad \hat{\boldsymbol{s}} \geq 0 \tag{31}$$

where $K = G^{-1}$ is the convolution matrix with entries $K_{t,u} = h_{1+t-u}$ if $t \geq u$ else zero; the kernel vector $\boldsymbol{h}$ can be taken as an arbitrary response kernel for most of the development in this section. As noted earlier, $h_{1+t} = (A^t)_{1,1}$ for the special case of an AR process. As we have seen previously, the effect of the sparsity penalty (together with the non-negative constraint) is to shift the data down by a vector $\boldsymbol{\mu} = \lambda K^{-\top} \mathbf{1}$, and the problem reduces to a non-negative least squares (NNLS) problem.

$$\underset{\hat{\boldsymbol{s}}}{\text{minimize}} \quad \frac{1}{2} \| K\hat{\boldsymbol{s}} - (\boldsymbol{y} - \lambda K^{-\top} \mathbf{1}) \|^2 \quad \text{subject to} \quad \hat{\boldsymbol{s}} \geq 0. \tag{32}$$

(Note that the gradient of Eq (32) is the same as the gradient of Eq (31), $K^\top (K\hat{\boldsymbol{s}} - (\boldsymbol{y} - \lambda K^{-\top} \mathbf{1})) = K^\top (K\hat{\boldsymbol{s}} - \boldsymbol{y}) + \lambda \mathbf{1}$. In addition, K is triangular with positive numbers on the main diagonal, hence $\det K > 0$ and K is invertible.)

A classic algorithm for solving a NNLS problem is the active set method of [37] and [38]. This algorithm alternates between normal equation matrix solves involving sub-matrices of $K^\top K$ and updates of the active set. A naive application of this algorithm would scale cubically with the number of spikes. Instead, we exploit the locality of the problem (the fact that changing a spike height at time t does not affect the solution at very distant times s) and apply the

NNLS algorithm in the inner loop of a sequential coordinate block descent method. Specifically, we apply warm-started NNLS on blocks of size Δ (where Δ is the length of the calcium transient), stepping the block in steps of size $\Delta_m < \Delta$ (we found $\Delta_m = \frac{\Delta}{2}$ to be effective for offline applications; for online applications Δ_m would be set smaller) and applying NNLS while holding the values of s outside the block fixed. We further exploit the Toeplitz structure of K to precompute the necessary sub-matrices of $K^\top K$.

Algorithm 3 Fast online deconvolution for arbitrary convolution kernels

Require: kernel $\boldsymbol{h}$, regularization parameter λ, window size Δ, shift size Δ_m, data subset $\boldsymbol{y}_{t:t+\Delta-1} \subset \boldsymbol{y}$ at time of reading

initialize $K_{t,u} \leftarrow h_{1+t-u}$ for $1 \le u \le t \le \Delta$, $\boldsymbol{y} \leftarrow \boldsymbol{y} - \lambda K^{-\top}\mathbf{1}$, $A \leftarrow K^\top K$, $t \leftarrow 1$

while $t + \Delta \le T$ **do**

  $\hat{\boldsymbol{s}}_{t:t+\Delta-1} \leftarrow \mathrm{NNLS}\,(A, K^\top \boldsymbol{y}_{t:t+\Delta-1}, \hat{\boldsymbol{s}}_{t:t+\Delta-1})$ ▷classic NNLS on $\hat{\boldsymbol{s}}_{t:t+\Delta-1}$, but warm-started†

  $\boldsymbol{y}_{t:t+\Delta-1} \leftarrow \boldsymbol{y}_{t:t+\Delta-1} - K_{:,1:\Delta_m}\hat{\boldsymbol{s}}_{t:t+\Delta_m-1}$ ▷peel off contribution of previous activity

  $t \leftarrow t + \Delta_m$

$\hat{\boldsymbol{s}}_{t:T} \leftarrow \mathrm{NNLS}\,(A_{t+\Delta-T:\Delta,t+\Delta-T:\Delta}, K^\top_{1:T-t+1,1:T-t+1}\boldsymbol{y}_{t:T}, \hat{\boldsymbol{s}}_{t:T})$ ▷robustness to $\frac{T-\Delta}{\Delta_m} \notin \mathbb{N}$

return $\hat{\boldsymbol{s}}$

†The function NNLS implements a minor variation of the classic algorithm of [38] to solve $\min_{\hat{\boldsymbol{s}} \in \mathbb{R}^T_+} \|\boldsymbol{y} - K\hat{\boldsymbol{s}}\|^2$: $K^\top K$ and $K^\top \boldsymbol{y}$ are precomputed outside the function, to exploit that NNLS is called several times with the same K. Further $\hat{\boldsymbol{s}}$ is warm-started instead of initializing it as $\mathbf{0}$.

The resulting algorithm (Alg 3) runs in $O(T)$ time. It involves solving a least squares problem for the time points within the considered window where $\hat{s}_t > 0$; thus it scales cubically with the number of spikes per window and depends on the sparsity of $\hat{\boldsymbol{s}}$. (In fact, for AR(p) models, the required matrix solves can be performed using linear-time (not cubic-time) Kalman filter-smoother methods, but the matrix sizes were sufficiently small in the examples examined here that the Kalman implementation was not necessary.) Further speedups can be obtained by restricting the set of possible spike times, for example, by running the AR(1) version of OASIS on a temporally decimated version of the signal to crudely identify the set of spike times, then never updating $\hat{\boldsymbol{s}}$ away from zero on the complement of this set.

To summarize, we describe in Algorithm 4 how the algorithmic variants introduced here are combined into a final full algorithm that includes hyperparameter optimization, the variants for AR(1) or AR(2), and soft (ℓ_1 penalty) or hard shrinkage (ℓ_0 penalty).

Algorithm 4 Full algorithm with hyperparameter optimization

Require: data $\boldsymbol{y}$, order p of the AR-process, sparsity norm q

1: initialize
2: AR parameters $\hat{\gamma}_1, \ldots, \hat{\gamma}_p$ using autocorrelation of $\boldsymbol{y}$
3: noise level $\hat{\sigma}$ using PSD of $\boldsymbol{y}$
4: background $\hat{b}$ using percentile of $\boldsymbol{y}$
5: dual variable $\lambda \leftarrow 0$
6: $\tilde{\boldsymbol{y}} \leftarrow$ temporally decimate batch of $\boldsymbol{y}$ ▷for faster hyperparameter optimization
7: rescale hyperparamaters due to decimation
8: **while** hyperparamaters not converged **do** ▷optimize hyperparameters, cf. Fig 3
9: Run warm-started Alg 2 on $\tilde{\boldsymbol{y}}$ with current hyperparameters
10: Update hyperparameters ▷Eqs (19, 23 and 24)
11: **if** $q = 0$ **then** determine $s_{\min}$ ▷Sec. 'Hard shrinkage and ℓ_0 penalty'
12: rescale hyperparamaters using the inverse transformations of line 7
13: $\hat{\boldsymbol{c}}, \hat{\boldsymbol{s}} \leftarrow$ run Alg 2 on full data $\boldsymbol{y}$

```
if p = 1 then
  return ŝ
else
  ŝ ← run warm-started Alg 3 on full data y
  return ŝ
```

Results

Benchmarking OASIS

We generated datasets of 20 fluorescence traces each for $p = 1$ and 2 with a duration of 100 s at a framerate of 30 Hz, such that $T = 3{,}000$ frames. The spiking signal came from a homogeneous Poisson process. We used $\gamma = 0.95$, $\sigma = 0.3$ for the AR(1) model and $\gamma_1 = 1.7$, $\gamma_2 = -0.712$, $\sigma = 1$ for the AR(2) model. Fig 4A–4C are reassuring that our suggested (dual) active set method yields indeed the same results as other convex solvers for an AR(1) process and that spikes are extracted well. For an AR(2) process OASIS is greedy and yields good results that are similar to

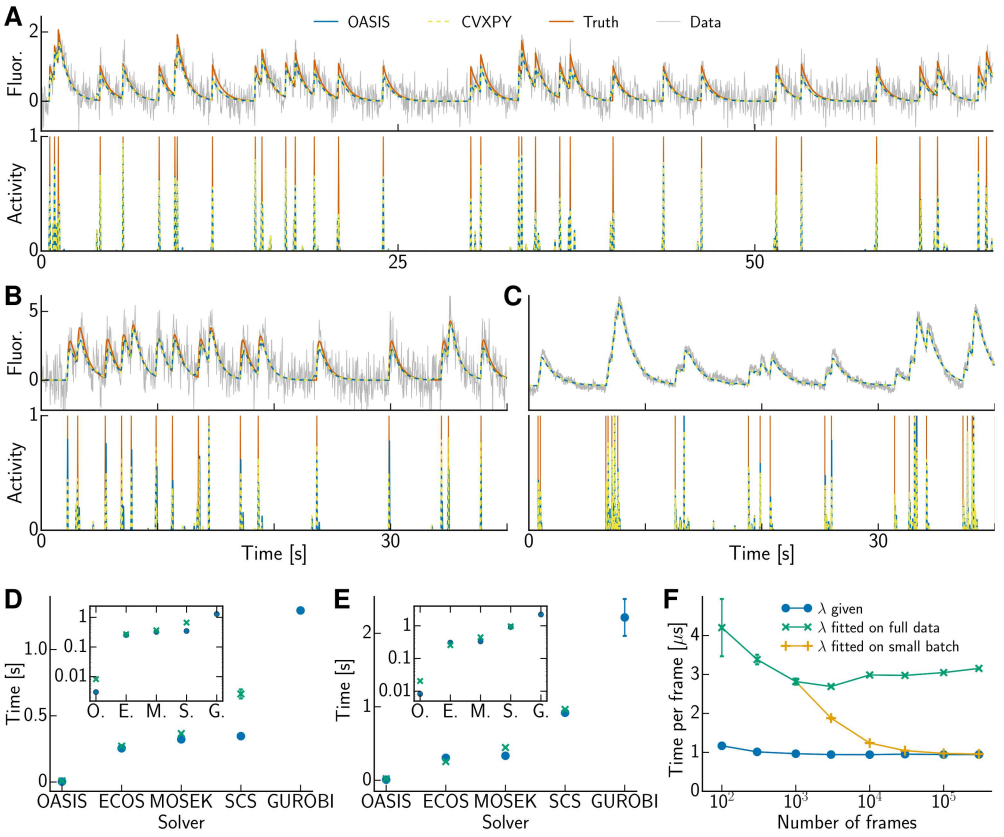

Fig 4. OASIS produces the same high quality results as convex solvers at least an order of magnitude faster. (A) Raw and inferred traces for simulated AR(1) data, **(B)** simulated AR(2) and **(C)** real data from [36] fitted with an AR(2) model. OASIS solves Eq (3) exactly for AR(1) and just approximately for AR(2) processes, nevertheless well extracting spikes. **(D)** Computation time for simulated AR(1) data with given λ (blue circles, Eq 3) or inference with hard noise constraint (green x, Eq 15). GUROBI failed on the noise constrained problem. The inset shows the same data in logarithmic scale. **(E)** Computation time for simulated AR(2) data. **(F)** Normalized computation time of OASIS for simulated AR(1) data with given λ (blue circles, Eq 3) and inference with hard noise constraint on full data (green x, Eq 15) or small initial batch followed by processing in online mode (orange crosses).

https://doi.org/10.1371/journal.pcbi.1005423.g004

the one obtained with convex solvers (lower panels in Fig 4B and 4C), with virtually identical denoised fluorescence traces (upper panels).

Fig 4D and 4E report the computation time (±SEM) averaged over all 20 traces and ten runs per trace on a MacBook Pro with Intel Core i5 2.7 GHz CPU. We compared the run time of our algorithm to a variety of state of the art convex solvers that can all be conveniently called from the convex optimization toolbox CVXPY [39]: embedded conic solver (ECOS, [40]), MOSEK [41], splitting conic solver (SCS, [42]) and GUROBI [43]. ECOS and MOSEK are the most competitive methods; these are interior-point methods that cannot use warm starts. With a known sparsity parameter λ (Eq 3), OASIS is about two magnitudes faster than any other method for an AR(1) process (Fig 4D, blue disks) and more than one magnitude for an AR(2) process (Fig 4E). Whereas several of the other solvers take almost the same time for the noise constrained problem (Eq 15, Fig 4D and 4E, green x), our method takes about three times longer to find the value of the dual variable λ compared to the formulation where the residual is part of the objective; nevertheless it still outperforms the other algorithms by a huge margin.

We also ran the algorithms on longer traces up to a length of $T = 300{,}000$ frames (Fig 4F), confirming that OASIS scales linearly with T, where we obtained a proportionality constant of 1 μs/frame. For an unknown hyperparameter λ we obtained its value not only on the full data but on an initial small batch (1,000 frames) and kept it fixed, which sped activity inference up by a factor of three once T is sufficiently large (Fig 4F, orange vs green) without compromising quality (correlation between *deconvolved* activity and ground truth spike train 0.882 ± 0.001 vs 0.881 ± 0.002 for $T = 300{,}000$). Our active set method maintained its lead by 1–2 orders of magnitude in computing time. Further, compared to our active set method the other algorithms required at least an order of magnitude more RAM, confirming that OASIS is not only faster but much more memory efficient. Indeed, because OASIS can run in online mode the memory footprint can be $O(1)$, instead of $O(T)$.

We verified these results on real data as well. Running OASIS with the hard noise constraint and $p = 2$ on the GCaMP6s dataset of 14,400 frames collected at 60 Hz from [36, 44] took 0.101±0.005 s per trace, whereas the fastest other methods required 2.37±0.12 s. Fig 4C shows the real data together with the inferred denoised and deconvolved traces as well as the true spike times, which were obtained by simultaneous electrophysiological recordings [36].

We also extracted each neuron's fluorescence activity using CNMF from an unpublished whole-brain zebrafish imaging dataset from the M. Ahrens lab. Running OASIS with hard noise constraint and $p = 1$ (chosen because the calcium onset was fast compared to the acquisition rate of 2 Hz) on 10,000 traces out of a total of 91,478 suspected neurons took 81.5 s whereas ECOS, the fastest competitor, needed 2,818.1 s. For all neurons we would hence expect 745 s for OASIS, which is below the 1,500 s recording duration (3,000 frames), and over 25,780 s for ECOS and other candidates.

OASIS solves the non-negative deconvolution problem exactly for an AR(1) process; however, as discussed above, for $p > 1$ the solution is only a good (greedy) approximation. To obtain the exact solution we ran the ONNLS algorithm on the simulated AR(2) traces using a window size of 200 frames, which was about ten times larger than the fluorescence decay time, and shifting the window by 100 frames. We obtained higher accuracy results than all the state of the art convex solvers we compared to, requiring merely 27.8±0.4 ms per trace for $\lambda = 0$ and 20.0±0.4 ms per trace for $\lambda = 30$, the value that ensures that the hard noise constraint is tight. The choice of λ regulated the sparsity of the solution, which affects the run time of ONNLS. The fastest state of the art convex solver (ECOS) required 305±9 ms and was thus an order of magnitude slower. It took merely 8.56±0.04 ms to obtain an approximate greedy solution using OASIS, independent of the choice of sparsity parameter λ. Though obtaining the exact

solution requires more computing time, it is well within the same order of magnitude. In contrast, running batch NNLS was significantly slower, requiring 2,430±53 ms for $\lambda = 0$ and 1,620±37 ms for $\lambda = 30$. Solving the noise constrained problem by iterating warm-started ONNLS to obtain the corresponding value of the dual variable λ took 73±1 ms. However, we can improve on that by first running the fast but (for $p > 1$) approximate dual method to obtain a good estimate of λ as well as $\boldsymbol{s}$, and then switching to the slower but exact primal method. Running OASIS and executing warm-started ONNLS just once required collectively merely 23±1 ms, similarly to cold-started ONNLS with given λ. Running ONNLS not just once, but until the value of λ has been further tuned such that the noise constraint holds not approximately but exactly, took altogether 31±1 ms.

Hyperparameter optimization

We have shown that we can solve Eqs (3) and (15) faster than interior point methods. The AR coeffient γ was either known or estimated based on the autocorrelation in the above analyses. The latter approach assumes that the spiking signal comes from a homogeneous Poisson process, which does not generally hold for realistic data. Therefore we were interested in optimizing not only the sparsity parameter λ, but also the AR(1) coeffient γ. To illustrate the optimization of both, we generated a fluorescence trace with spiking signal from an inhomogeneous Poisson process with sinusoidal instantaneous firing rate (Fig 3). We conservatively initialized $\hat{\gamma}$ to a small value of 0.9. The value obtained based on the autocorrelation was 0.9792 and larger than the true value of 0.95. The left panels in Fig 3B and 3D illustrate the update of λ from the previous value λ^- to λ^* by solving a quadratic equation analytically (Eq 18) and the update of $\hat{\gamma}$ by numerical minimization of a high order polynomial respectively. Note that after merely one iteration (Fig 3E) a good solution is obtained and after three iterations the solution is virtually identical to the one obtained when the true value of γ has been provided (Fig 3F). This holds not only visually, but also when judged by the correlation between *deconvolved* activity and ground truth spike train, which was 0.869 compared to merely 0.773 if $\hat{\gamma}$ was obtained based on the autocorrelation. The optimization was robust to the initial value of $\hat{\gamma}$, as long as it was positive and not, or only marginally, greater than the true value. The value obtained based on the autocorrelation was considerably greater and partitioned the time series into pools in a way that missed entire spikes.

After illustrating the hyperparameter optimization we next quantify the computing time and quality of spike inference for various optimization scenarios. We generated 20 fluorescence traces with sinusoidal instantaneous firing rate as used in the illustration (Fig 3), again having a duration of 100 s at a framerate of 30 Hz, such that $T = 3{,}000$ frames, however we offset the data by an additional positive baseline b that can be present in real data. This baseline can be optimized together with the sparsity parameter λ, as shown in Methods (subsection "Additional baseline"). The fastest deconvolution method is to merely estimate all parameters and run OASIS just once, cf. first row in Table 1 which shows the mean (±SEM) for computing time as well as correlation of the inferred spike train. As a baseline estimate we used the 15% percentile of the fluorescence trace. The sparsity penalty was set to $\lambda = 0$. A better choice of λ is actually obtained by optimizing it, such that the hard noise constraint $\|b\mathbf{1} + \hat{\boldsymbol{c}} - \boldsymbol{y}\|^2 = \hat{\sigma}^2 T$ holds, cf. second row in Table 1. The next rows show that optimizing b further improves the result, as does adding γ. However, the increased number of optimized parameters results in extra computational cost. The computation time can be reduced by estimating γ not using the full data but only a limited number of pools, which does not affect the quality of the result, cf. row five and six in Table 1. Note that by restricting the optimization to a fixed number of pools, its computational load does not increase with the duration of the recording, hence the

Table 1. Cost and quality of spike inference with parameter optimization.

optimize	accelerate	Time [ms]	Correlation
-	-	3.25 ± 0.03	0.831 ± 0.006
λ	-	9.2 ± 0.1	0.849 ± 0.006
λ, *b*	-	8.4 ± 0.2	0.857 ± 0.007
λ, *b*, *γ*	-	48.4 ± 2.3	0.875 ± 0.006
λ, *b*, *γ*	use 10 pools	16.0 ± 0.4	0.875 ± 0.006
λ, *b*, *γ*	use 5 pools	14.2 ± 0.3	0.875 ± 0.006
λ, *b*, *γ*	decimate	29.4 ± 1.3	0.878 ± 0.006
λ, *b*, *γ*	decimate, use 10 pools	12.1 ± 0.2	0.878 ± 0.006
λ, *b*, *γ*	decimate, use 5 pools	10.6 ± 0.2	0.877 ± 0.006

The first column shows the quantities that have been optimized, the second methods used to accelerate the parameter optimization, the third the computing time per trace (±SEM) and the last shows the performance of spike train inference using the correlation between inferred activity and true spike train. We used 20 simulated fluorescence traces with a spiking signal coming from an inhomogeneous Poisson process and a duration of 100 s at a framerate of 30 Hz such that $T = 3{,}000$ frames.

https://doi.org/10.1371/journal.pcbi.1005423.t001

gain would be even more dramatic for longer time series. Further speed ups are obtained by estimating the parameters on a decimated version of the data, as the last rows in Table 1 illustrate. Here we decimated the fluorescence traces by a factor of ten, without harming the inference quality.

Hard thresholding

OASIS solves a LASSO problem resulting in soft shrinkage. The deconvolved trace $\hat{\boldsymbol{s}}$ typically has values smaller than 1 and often shows "partial spikes" in neighboring bins reflecting the uncertainty regarding the exact position of a spike, cf. Fig 4. While this information can be useful, one sometimes wants to merely commit to one event within a time bin instead and get rid of remaining small values in $\hat{\boldsymbol{s}}$. We ran a slightly modified version of the algorithm that replaces the sparsity penalty by a constraint on the minimal spike size $s_{\min}$, yielding sparser solutions but rendering the problem non-convex. Although we are not guaranteed to find the global minimum, we obtained good results, cf. Fig 5. To quantify directly the similarity

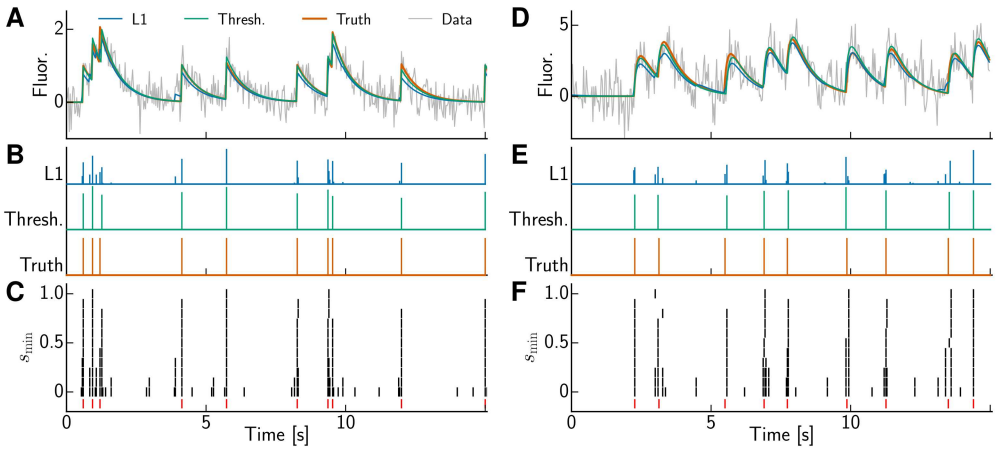

Fig 5. Thresholding can improve the accuracy of spike inference. (A) Inferred trace using L1 penalty (L1, blue) and the thresholded OASIS (Thresh., green). The data (gray) are simulated with AR(1) model. **(B)** Inferred spiking activity. **(C)** The detected events using thresholded OASIS depend on the selection of $s_{\min}$. The ground truth is shown in red. **(D,E,F)**, same as **(A,B,C)**, but the data are simulated with AR(2).

https://doi.org/10.1371/journal.pcbi.1005423.g005

between the inferred deconvolved trace and ground truth spike train we calculated the correlation between the two. The best results were obtained for $s_{\min} = 0.5$ yielding correlation 0.899 ± 0.009 with the true spike train compared to 0.879 ± 0.006 for the solution of the problem with hard noise constraint (Eq 15). However, in a practical application the scaling factor between calcium fluorescence and a single spike, which is 1 for our simulated data, is often unknown, rendering it impossible to simply set the threshold $s_{\min}$ to the half of it. Instead, we can vary the threshold until the RSS crosses the threshold $\sigma^2 T$. The order in which the pools are merged or split matters for this non-convex case and sequentially adding spikes at the highest values of the ℓ_1-solution yielded the best performance with correlation 0.888 ± 0.007.

Fig 5 also shows results with a constraint on the minimal spike size for an AR(2) process. Adding the constraint helps when pressed for a binary decision whether to assign a spike or not, yielding visually excellent results. However, with a finite rise time of the calcium response the onset detection is notoriously difficult, because for a low threshold there are a lot of false positives due to noise, whereas for a high threshold, closer to the peak of the calcium kernel, the onset has already occurred earlier. Indeed, the greedy method for an AR(2) process tends to register spikes too late, which is further exacerbated when a threshold on the spike size ($s_{\min} = 0.5$) is introduced, leading to low values of spike similarity (correlation 0.419 ± 0.016) compared to the solution of basis pursuit denoising (Eq 15) (correlation 0.497 ± 0.013). We can incorporate a correction step that whenever a new spike is added, slightly jitters the previous one and calculates the change in the optimization objective in order to determine the optimal placement of the spike. For simplicity and low computational burden, we restrict the consideration of the changing RSS to the pools prior and after the jittered spike, which improves the spike detection (correlation 0.462 ± 0.015) while only marginally increasing computational cost (from 8.65 ms to 11.65 ms). Further improvements can be obtained by following up with (O)NNLS. The solution obtained by OASIS with threshold on the minimal spike size and jittering can be used to restrict (O)NNLS to have non-zero values only in close proximity to the spikes of the greedily obtained solution. This processing step increased the performance of spike inference to correlation 0.530 ± 0.010, which is better than the already mentioned one obtained for exactly solving the convex problem (Eq 15). Hence, though imposing a minimal spike size renders the problem non-convex, a tractable approximate solution to this problem can improve over the exact solution of the convex basis pursuit denoising problem.

In the AR(2) case the exact solutions (ONNLS with λ or ONNLS with support only in the proximity of the thresholded solution) consistently improved over the faster greedy methods, as measured by spike train correlation. The performance was hardly affected by whether the penalized or the thresholded version was chosen. Spike train correlation harshly penalizes spikes that are detected but at an incorrect time, no matter how close; therefore the activity plots and correlation values convey somewhat complementary information about the quality of the inference. We attribute the performance gap between greedy and exact solutions to greedy methods missing the exact time step more often. However, the optimally attainable time resolution is already limited by low SNR, in particular if the rise time of the calcium indicator is finite. Indeed, being more lenient regarding the exact spike timing we calculated the correlations after convolving the spike trains with a Gaussian with standard deviation of one bin width. The correlation values increased to 0.731 ± 0.008 for the greedy thresholded solution and to 0.800 ± 0.007 if followed up by ONNLS, but did not increase further for wider Gaussian kernels. This indicates that in the considered SNR regime single time bin resolution is out of reach, but spike times can be inferred with an uncertainty of about one time bin width.

Online spike inference with limited lag

For an exact solution of the non-negative deconvolution problem of an AR(1) process OASIS needs to backtrack to the most recent spike. (For an AR(2) process the solution is greedy and merely approximate. ONNLS yields an exact solution in this case but considers an even wider time window.) Such delays could be too long for some interesting closed loop experiments; therefore we were interested in how well the method performs if backtracking is limited to just a few frames. We varied the lag in the online estimator, i.e. the number of future samples observed before assigning a spike at time zero, for different signal-to-noise ratios (SNR). For each lag we chose the sparsity parameter λ such that the noise constraint $\|\hat{\boldsymbol{c}} - \boldsymbol{y}\|^2 \leq \sigma^2 T$ was tight. This yielded increasing values of λ for smaller lags, compensating for the fact that limiting backtracking to fewer frames also imposes fewer constraints ($\hat{c}_t \geq \gamma \hat{c}_{t-1}$) on the dynamics. In the case of hard thresholding, better results were obtained with higher $s_{\min}$ for smaller lags too, in order to avoid that one spike is split in two. We used a hand-chosen value of $s_{\min} = 0.5 + 0.175\, e^{-\tau}$ where τ is the lag, that asymptotically approaches the 0.5 for batch processing. The obtained results are depicted in Fig 6. For realistic SNR (3–5, though [36] report even higher values, cf. Fig 4C) and sample rates (30 Hz), lags of 2–5 yielded virtually the same results as offline estimation. The exact number depends on the noise; however, the main effect of noise was to reduce the optimal performance attainable even with batch processing, as the asymptotic values in Fig 6A and 6B reveal.

Discussion

We presented an online active set method for spike inference from calcium imaging data. We assumed that the forward model to generate a fluorescence trace from a spike train is linear-Gaussian. Further development will extend the method to nonlinear models [45] incorporating saturation effects and a noise variance that increases with the mean fluorescence to better resemble the Poissonian statistics of photon counts. In S1 Appendix we already extend our mathematical formulation to include weights for each time point as a first step in this direction.

Our method considered spike inference as a sparse non-negative deconvolution problem. We focused on the formulation that imposes sparsity using an ℓ_1 penalty that renders the

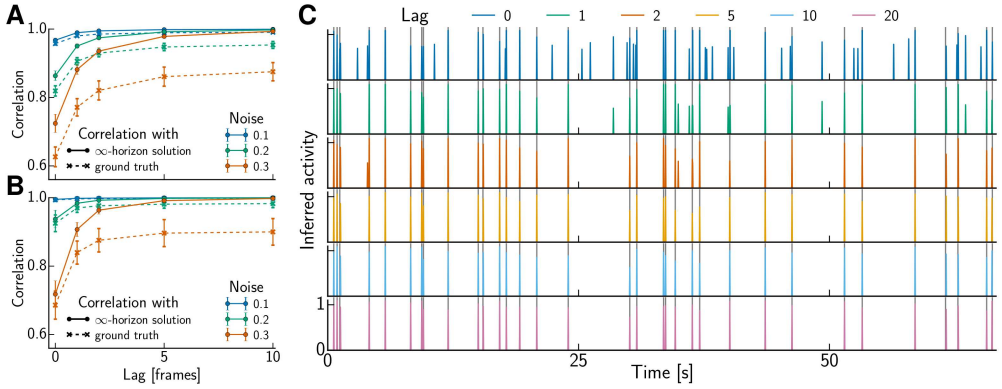

Fig 6. Varied lag in the online estimator. (A,B) Performance of spike inference as function of lag for various noise levels (i.e., inverse SNR) without (A) and with positive threshold $s_{\min}$ (B). We used correlation of the inferred spike train as similarity measure and compared to ground truth as well as to the optimally recoverable activity when the lag is unlimited as in offline processing. **(C)** Inferred trace with positive threshold $s_{\min}$ for increasing lag using the data depicted in Fig 4A with high noise level ($\sigma = 0.3$). The gray lines indicate the true spike times.

https://doi.org/10.1371/journal.pcbi.1005423.g006

problem convex. Using this problem formulation for spike inference has already long standing success within the neuroscientific community. We were able to speed it up by an order of magnitude compared to previously employed interior point methods and derived an algorithm that lends itself to online applications. However, recently several investigators [46–48] have advocated sparser methods, e.g. by using an ℓ_q-norm with $q < 1$ instead of $q = 1$ [46] or by enforcing refractoriness [47] (see also [13] for some further discussion of sparsening beyond ℓ_1 penalization). They report improved results, however in some cases at the expense of non-convexity, thus losing the guarantee of finding the global optimum. We leave it to future work to incorporate refractoriness into the methods developed here, but we did slightly modify the sparse non-negative deconvolution problem by adding the constraint that positive spikes need to be larger than some minimal value. A minor modification to our algorithm enabled it to find an (approximate) solution of this non-convex problem, which can be marginally better than the solution obtained with ℓ_1 regularizer. The ℓ_1-penalized solution reflects the uncertainty regarding the exact position of a spike by distributing it as "partial spikes" over neighboring bins. The thresholded solution lets go of this potentially useful information and instead commits to one event within the locally optimal time bin. We leave it up to the user which approach to choose.

Availability

We provide Python and MATLAB implementations of our algorithm online (https://github.com/j-friedrich/OASIS and linked repositories therein). The code is readily usable on new data and includes example scripts that produce all figures and Table 1 of this article.

Here we focused on temporal data, i.e. noisy neural fluorescence data that has been extracted and demixed from raw pixel data. We further added OASIS as deconvolution subroutine to CaImAn (https://github.com/simonsfoundation/CaImAn) [49], which implements CNMF for simultaneous denoising, deconvolution, and demixing of spatio-temporal calcium imaging data.

Supporting information

S1 Appendix. Technical appendix. The supplementary material includes a naïve isotonic regression algorithm without pooling. We generalize OASIS to the case of weighted regression and provide a mathematical proof for updates according to Eqs (12–14). We further discuss how to account for elevated initial calcium fluorescence levels and provide explicit expressions of the hyperparameter updates for an AR(2) model.
(PDF)

S1 Video. Illustration of PAVA. The supplementary video illustrates PAVA. The pool currently under consideration is indicated by the blue crosses. The algorithm sweeps through the time series and enforces the order constraints $x_1 \leq \ldots \leq x_T$.
(MP4)

S2 Video. Illustration of OASIS. The supplementary video illustrates OASIS for an AR(1) process. As in Fig 2, red lines depict true spike times and the shaded background shows how the time points are gathered in pools. The pool currently under consideration is indicated by the blue crosses. The upper panel shows how the calcium fluorescence trace $\boldsymbol{c}'$ develops while the algorithm runs, cf. Fig 2. The video additionally shows the deconvolved trace $\boldsymbol{s}' = G\boldsymbol{c}'$ (Eq 3) in the lower panel. The algorithm sweeps through the time series and enforces the constraint $\boldsymbol{s}' \geq 0$.
(MP4)

Acknowledgments

We would like to thank Misha Ahrens and Yu Mu for providing whole-brain imaging data of larval zebrafish. We thank John Cunningham and Eftychios Pnevmatikakis for helpful conversations as well as Scott Linderman and Daniel Soudry for valuable comments on the manuscript.

Part of this work was previously presented at the Thirtieth Annual Conference on Neural Information Processing Systems (NIPS, 2016) [50].

Author Contributions

Conceptualization: JF LP.

Formal analysis: JF LP PZ.

Funding acquisition: JF LP.

Investigation: JF PZ.

Methodology: JF LP PZ.

Software: JF PZ.

Supervision: LP.

Writing – original draft: JF LP PZ.

Writing – review & editing: JF LP PZ.

References

1. Grienberger C, Konnerth C. Imaging calcium in neurons. Neuron. 2012; 73(5):862–885. https://doi.org/10.1016/j.neuron.2012.02.011 PMID: 22405199
2. Grewe BF, Langer D, Kasper H, Kampa BM, Helmchen F. High-speed in vivo calcium imaging reveals neuronal network activity with near-millisecond precision. Nat Methods. 2010; 7(5):399–405. https://doi.org/10.1038/nmeth.1453 PMID: 20400966
3. Yaksi E, Friedrich RW. Reconstruction of firing rate changes across neuronal populations by temporally deconvolved Ca^{2+} imaging. Nat Methods. 2006; 3(5):377–383. https://doi.org/10.1038/nmeth874 PMID: 16628208
4. Holekamp TF, Turaga D, Holy TE. Fast three-dimensional fluorescence imaging of activity in neural populations by objective-coupled planar illumination microscopy. Neuron. 2008; 57(5):661–672. https://doi.org/10.1016/j.neuron.2008.01.011 PMID: 18341987
5. Vogelstein JT, Packer AM, Machado TA, Sippy T, Babadi B, Yuste R, et al. Fast nonnegative deconvolution for spike train inference from population calcium imaging. J Neurophysiol. 2010; 104(6):3691–3704. https://doi.org/10.1152/jn.01073.2009 PMID: 20554834
6. Vogelstein JT, Watson BO, Packer AM, Yuste R, Jedynak B, Paninski L. Spike inference from calcium imaging using sequential Monte Carlo methods. Biophys J. 2009; 97(2):636–655. https://doi.org/10.1016/j.bpj.2008.08.005 PMID: 19619479
7. Pnevmatikakis, EA, Merel, J, Pakman, A, Paninski, L. Bayesian spike inference from calcium imaging data. Asilomar Conference on Signals, Systems and Computers. 2013;.
8. Deneux T, Kaszas A, Szalay G, Katona G, Lakner T, Grinvald A, et al. Accurate spike estimation from noisy calcium signals for ultrafast three-dimensional imaging of large neuronal populations in vivo. Nat Commun. 2016; 7. https://doi.org/10.1038/ncomms12190
9. Sasaki T, Takahashi N, Matsuki N, Ikegaya Y. Fast and accurate detection of action potentials from somatic calcium fluctuations. J Neurophysiol. 2008; 100(3):1668–1676. https://doi.org/10.1152/jn.00084.2008 PMID: 18596182
10. Theis L, Berens P, Froudarakis E, Reimer J, Rosón MR, Baden T, et al. Benchmarking spike rate inference in population calcium imaging. Neuron. 2016; 90(3):471–482. https://doi.org/10.1016/j.neuron.2016.04.014 PMID: 27151639

11. Mishchencko Y, Vogelstein JT, Paninski L. A Bayesian approach for inferring neuronal connectivity from calcium fluorescent imaging data. Ann Appl Stat. 2011; p. 1229–1261. https://doi.org/10.1214/09-AOAS303
12. Picardo MA, Merel J, Katlowitz KA, Vallentin D, Okobi DE, Benezra SE, et al. Population-Level Representation of a Temporal Sequence Underlying Song Production in the Zebra Finch. Neuron. 2016; 90(4):866–876. https://doi.org/10.1016/j.neuron.2016.02.016 PMID: 27196976
13. Pnevmatikakis EA, Soudry D, Gao Y, Machado TA, Merel J, Pfau D, et al. Simultaneous Denoising, Deconvolution, and Demixing of Calcium Imaging Data. Neuron. 2016; 89(2):285–299. https://doi.org/10.1016/j.neuron.2015.11.037 PMID: 26774160
14. Grosenick L, Marshel JH, Deisseroth K. Closed-loop and activity-guided optogenetic control. Neuron. 2015; 86(1):106–139. https://doi.org/10.1016/j.neuron.2015.03.034 PMID: 25856490
15. Rickgauer JP, Deisseroth K, Tank DW. Simultaneous cellular-resolution optical perturbation and imaging of place cell firing fields. Nat Neurosci. 2014; 17(12):1816–1824. https://doi.org/10.1038/nn.3866 PMID: 25402854
16. Packer AM, Russell LE, Dalgleish HW, Häusser M. Simultaneous all-optical manipulation and recording of neural circuit activity with cellular resolution in vivo. Nat Methods. 2015; 12(2):140–146. https://doi.org/10.1038/nmeth.3217 PMID: 25532138
17. Clancy KB, Koralek AC, Costa RM, Feldman DE, Carmena JM. Volitional modulation of optically recorded calcium signals during neuroprosthetic learning. Nat Neurosci. 2014; 17(6):807–809. https://doi.org/10.1038/nn.3712 PMID: 24728268
18. Lewi J, Butera R, Paninski L. Sequential optimal design of neurophysiology experiments. Neural Comput. 2009; 21(3):619–687. https://doi.org/10.1162/neco.2008.08-07-594 PMID: 18928364
19. Park M, Pillow JW. Bayesian active learning with localized priors for fast receptive field characterization. In: Adv Neural Inf Process Syst; 2012. p. 2348–2356.
20. Shababo B, Paige B, Pakman A, Paninski L. Bayesian inference and online experimental design for mapping neural microcircuits. In: Adv Neural Inf Process Syst; 2013. p. 1304–1312.
21. Ahrens MB, Orger MB, Robson DN, Li JM, Keller PJ. Whole-brain functional imaging at cellular resolution using light-sheet microscopy. Nat Methods. 2013; 10(5):413–420. https://doi.org/10.1038/nmeth.2434 PMID: 23524393
22. Vladimirov N, Mu Y, Kawashima T, Bennett DV, Yang CT, Looger LL, et al. Light-sheet functional imaging in fictively behaving zebrafish. Nat Methods. 2014;. https://doi.org/10.1038/nmeth.3040
23. Potra FA, Wright SJ. Interior-point methods. J Comput Appl Math. 2000; 124(1):281–302. https://doi.org/10.1016/S0377-0427(00)00433-7
24. Ayer M, Brunk HD, Ewing GM, Reid WT, Silverman E. An empirical distribution function for sampling with incomplete information. Ann Math Stat. 1955; 26(4):641–647. https://doi.org/10.1214/aoms/1177728423
25. Barlow RE, Bartholomew DJ, Bremner JM, Brunk HD. Statistical inference under order restrictions: The theory and application of isotonic regression. Wiley New York; 1972.
26. van Eeden C. Testing and Estimating Ordered Parameters of Probability Distributions. PhD thesis, University of Amsterdam; 1958.
27. Miles RE. The complete amalgamation into blocks, by weighted means, of a finite set of real numbers. Biometrika. 1959; 46(3/4):317–327. https://doi.org/10.1093/biomet/46.3-4.317
28. Mair P, Hornik K, de Leeuw J. Isotone optimization in R: pool-adjacent-violators algorithm (PAVA) and active set methods. J Stat Softw. 2009; 32(5):1–24.
29. Best MJ, Chakravarti N. Active set algorithms for isotonic regression; a unifying framework. Math Prog. 1990; 47(1–3):425–439. https://doi.org/10.1007/BF01580873
30. Grotzinger SJ, Witzgall C. Projections onto order simplexes. Appl Math Optim. 1984; 12(1):247–270. https://doi.org/10.1007/BF01449044
31. Podgorski K, Haas K. Fast non-negative temporal deconvolution for laser scanning microscopy. J Biophotonics. 2013; 6(2):153–162. https://doi.org/10.1002/jbio.201100133 PMID: 22438321
32. Brent RP. Algorithms for Minimization Without Derivatives. Courier Corporation; 1973.
33. Nocedal J, Wright S. Numerical optimization. Springer Science & Business Media; 2006.
34. Donoho DL. De-noising by soft-thresholding. IEEE Trans Inf Theory. 1995; 41(3):613–627. https://doi.org/10.1109/18.382009
35. Mallat SG, Zhang Z. Matching pursuits with time-frequency dictionaries. IEEE Trans Signal Processing. 1993; 41(12):3397–3415. https://doi.org/10.1109/78.258082

36. Chen TW, Wardill T, Sun Y, Pulver SR, Renninger SL, Baohan A, et al. Ultrasensitive fluorescent proteins for imaging neuronal activity. Nature. 2013; 499(7458):295–300. https://doi.org/10.1038/nature12354 PMID: 23868258

38. Lawson CL, Hanson RJ. Solving least squares problems. vol. 15. SIAM; 1995.

37. Bro R, De Jong S. A fast non-negativity-constrained least squares algorithm. J Chemometrics. 1997; 11(5):393–401. https://doi.org/10.1002/(SICI)1099-128X(199709/10)11:5%3C393::AID-CEM483%3E3.3.CO;2-C

39. Diamond S, Boyd S. CVXPY: A Python-Embedded Modeling Language for Convex Optimization. J Mach Learn Res. 2016; 17(83):1–5.

40. Domahidi A, Chu E, Boyd S. ECOS: An SOCP solver for embedded systems. In: European Control Conference (ECC); 2013. p. 3071–3076.

41. Andersen ED, Andersen KD. The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In: High performance optimization. Springer; 2000. p. 197–232.

42. O'Donoghue B, Chu E, Parikh N, Boyd S. Conic Optimization via Operator Splitting and Homogeneous Self-Dual Embedding. J Optim Theory Appl. 2016; p. 1–27.

43. Gurobi Optimization Inc. Gurobi Optimizer Reference Manual; 2015. Available from: http://www.gurobi.com.

44. GENIE project, Janelia Research Campus, HHMI; Karel Svoboda (contact). Simultaneous imaging and loose-seal cell-attached electrical recordings from neurons expressing a variety of genetically encoded calcium indicators. CRCNS.org; 2015. Available from: http://dx.doi.org/10.6080/K02R3PMN.

45. Pologruto TA, Yasuda R, Svoboda K. Monitoring neural activity and [Ca^{2+}] with genetically encoded Ca^{2+} indicators. J Neurosci. 2004; 24(43):9572–9579. https://doi.org/10.1523/JNEUROSCI.2854-04.2004 PMID: 15509744

46. Quan T, Lv X, Liu X, Zeng S. Reconstruction of burst activity from calcium imaging of neuronal population via Lq minimization and interval screening. Biomed Opt Express. 2016; 7(6):2103–2117. https://doi.org/10.1364/BOE.7.002103 PMID: 27375930

47. Dyer EL, Studer C, Robinson JT, Baraniuk RG. A robust and efficient method to recover neural events from noisy and corrupted data. In: Int IEEE/EMBS Conf Neural Eng (NER). 2013; p. 593–596.

48. Pachitariu M, Stringer C, Schröder S, Dipoppa M, Rossi LF, Carandini M, et al. Suite2p: beyond 10,000 neurons with standard two-photon microscopy. bioRxiv. 2016.

49. Giovannucci A, Friedrich J, Deverett B, Staneva V, Chklovskii D, Pnevmatikakis E. CaImAn: An open source toolbox for large scale calcium imaging data analysis on standalone machines. Cosyne Abstracts 2017, Salt Lake City USA.

50. Friedrich J, Paninski L. Fast Active Set Methods for Online Spike Inference from Calcium Imaging. In: Adv Neural Inf Process Syst; 2016. p. 1984–1992.