RESEARCH ARTICLE

Online analysis of microendoscopic 1-photon calcium imaging data streams

Johannes Friedrich[1]*, **Andrea Giovannucci**[2], **Eftychios A. Pnevmatikakis**[1]

1 Flatiron Institute, Simons Foundation, New York, New York, United States of America, **2** Joint Department of Biomedical Engineering, University of North Carolina at Chapel Hill and North Carolina State University; and UNC Neuroscience Center, University of North Carolina at Chapel Hill, Chapel Hill, North Carolina, United States of America

* jfriedrich@flatironinstitute.org

Citation: Friedrich J, Giovannucci A, Pnevmatikakis EA (2021) Online analysis of microendoscopic 1-photon calcium imaging data streams. PLoS Comput Biol 17(1): e1008565. https://doi.org/10.1371/journal.pcbi.1008565

Editor: Daniel Bush, University College London, UNITED KINGDOM

Received: January 21, 2020

Accepted: November 27, 2020

Published: January 28, 2021

Data Availability Statement: The experimental data has been published previously and can be downloaded from https://doi.org/10.5061/dryad.kr17k. The simulated data is available at https://caiman.flatironinstitute.org/~neuro/caiman_downloadables/sim.mat.

Funding: JF, AG, and EAP were internally funded by and received their salary from the Simons Foundation (https://simonsfoundation.org). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Abstract

In vivo calcium imaging through microendoscopic lenses enables imaging of neuronal populations deep within the brains of freely moving animals. Previously, a constrained matrix factorization approach (CNMF-E) has been suggested to extract single-neuronal activity from microendoscopic data. However, this approach relies on offline batch processing of the entire video data and is demanding both in terms of computing and memory requirements. These drawbacks prevent its applicability to the analysis of large datasets and closed-loop experimental settings. Here we address both issues by introducing two different online algorithms for extracting neuronal activity from streaming microendoscopic data. Our first algorithm, OnACID-E, presents an online adaptation of the CNMF-E algorithm, which dramatically reduces its memory and computation requirements. Our second algorithm proposes a convolution-based background model for microendoscopic data that enables even faster (real time) processing. Our approach is modular and can be combined with existing online motion artifact correction and activity deconvolution methods to provide a highly scalable pipeline for microendoscopic data analysis. We apply our algorithms on four previously published typical experimental datasets and show that they yield similar high-quality results as the popular offline approach, but outperform it with regard to computing time and memory requirements. They can be used instead of CNMF-E to process pre-recorded data with boosted speeds and dramatically reduced memory requirements. Further, they newly enable online analysis of live-streaming data even on a laptop.

Author summary

Calcium imaging methods enable researchers to measure the activity of genetically-targeted large-scale neuronal subpopulations. Whereas previous methods required the specimen to be stable, e.g. anesthetized or head-fixed, new brain imaging techniques using microendoscopic lenses and miniaturized microscopes have enabled deep brain imaging in freely moving mice. However, the very large background fluctuations, the inevitable movements and distortions of imaging field, and the extensive spatial overlaps of fluorescent signals complicate the goal of efficiently extracting accurate estimates of neural

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

activity from the observed video data. Further, current activity extraction methods are computationally expensive due to the complex background model and are typically applied to imaging data long after the experiment is complete. Moreover, in some scenarios it is necessary to perform experiments in real-time and closed-loop—analyzing data on-the-fly to guide the next experimental steps or to control feedback –, and this calls for new methods for accurate real-time processing. Here we address both issues by adapting a popular extraction method to operate online and extend it to utilize GPU hardware that enables real time processing. Our algorithms yield similar high-quality results as the original offline approach, but outperform it with regard to computing time and memory requirements. Our results enable faster and scalable analysis, and open the door to new closed-loop experiments in deep brain areas and on freely-moving preparations. Our algorithms can be used for newly enabled real-time analysis of streaming data, as well as swapped in directly to replace the computationally costly offline approach.

Introduction

In vivo calcium imaging of activities from large neural populations at single cell resolution has become a widely used technique among experimental neuroscientists. Recent advances in optical imaging technology using a 1-photon-based miniscope and a microendoscopic lens have enabled in vivo calcium imaging studies of neural activities in freely behaving animals [1–4]. However, this data typically displays large, highly structured background fluctuations due to fluorescence contributions from neurons outside the focal plane, arising from the large integration volume of one photon microscopy. To obtain a robust approach for extracting single-neuronal signals from microendoscopic data the constrained nonnegative matrix factorization (CNMF, [5]) approach has been extended to leverage a more accurate and flexible spatio-temporal background model, able to capture the properties of the strong background signal (CNMF-E, [6]). This prevalent algorithm (see [7] for an alternative proposal) has been widely used to study neural circuits in cortical and subcortical brain areas, e.g. prefrontal cortex (PFC, [8]) and hippocampus [2, 9], as well as previously inaccessible deep brain areas, such as striatum [10, 11], amygdala [12], substantia nigra pars compacta (SNc) [13], nucleus accumbens [14], dorsolateral septum [15], parabrachial nucleus [16], and other brain regions.

A concomitant feature of the refined background model in CNMF-E is its high computational and memory cost. Although the data can be processed by splitting and processing the FOV in smaller patches to exploit a time/memory tradeoff [17], this strategy requires significant time resources and does not scale to longer recordings. Further, CNMF-E is 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 [18–20]—and this requires new methods for accurate real-time processing.

Online (and real time) analysis of calcium imaging data has been proposed with the OnACID algorithm [21]. The algorithm combines the online NMF algorithm of [22], the CNMF source extraction algorithm of [5], and the near-online deconvolution algorithm of [23], to provide an automated pipeline that can discover and track the activity of hundreds of cells in real time, albeit only for 2-photon or light-sheet imaging data.

In this paper, we present two algorithms for the online analysis of microendoscopic 1-photon calcium imaging data streams. Our first algorithm (OnACID-E), extends [21] by incorporating the background model and neuron detection method of CNMF-E [6] and adapting them to an online setup. Our second approach proposes a lower dimensional background model by introducing parameter sharing through a convolutional structure and combines it with the online 2-photon processing of [21]. In either approach, every frame is processed in four sequential steps: i) The frame is registered against the previous background-corrected denoised frame to correct for motion artifacts. ii) The fluorescence activity of the already detected sources is tracked. iii) Newly appearing neurons and processes are detected and incorporated to the set of existing sources. iv) The fluorescence trace of each source is denoised and deconvolved to provide an estimate of the underlying spiking activity.

Our resulting framework is highly scalable with minimal memory requirements, as it processes the data in streaming mode (one frame at a time), while keeping in memory a set of low dimensional sufficient statistics and a small minibatch of the most recent data frames. Moreover, it results in faster processing that can reach real time speeds for common experimental scenarios. We apply our framework to typical mouse *in vivo* microendoscopic 1p datasets; our algorithm can find and track hundreds of neurons faster than real-time, and outperforms the CNMF-E algorithm of [6] with regard to computing time and memory requirements while maintaining the same high quality of the results. We also provide a Python implementation of our methods as part of the CaImAn package [17].

Methods

This section is organized as follows. The first subsection briefly reviews the modeling assumptions of CNMF-E for microendoscope data. In the second subsection, we derive an online method to fit this model, thus enabling the processing of 1-photon endoscopic data streams (OnACID-E). In the third subsection, we modify the background modeling assumptions to introduce a convolutional structure and describe how to utilize this to derive an alternative fast online algorithm. Finally, we describe how motion correction, which is typically done as preprocessing step, can be performed online, and stream processing allows us to employ a very simple yet effective motion correction scheme. Throughout we use the common convention to denote vectors and matrices with boldface lowercase and uppercase letters respectively. We use i, j for general indexing. Where it helps the exposition, we use a different lowercase letter as index and the corresponding uppercase letter as its upper bound, e.g. $t \in \{1, \ldots, T\}$ as time index where T is the total number of frames observed, and $n \in \{1, \ldots, N\}$ as neuron index where N is the total number of neurons.

CNMF for microendoscopic data (CNMF-E)

The recorded video data can be represented by a matrix $\boldsymbol{Y} \in \mathbb{R}_+^{d\times T}$, where d is the number of imaged pixels and T is the number of frames observed. Following [5], we model $\boldsymbol{Y}$ as

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{C} + \boldsymbol{B} + \boldsymbol{E}, \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}_+^{d\times N}$ is a spatial matrix that encodes the location and shape of each neuron (spatial footprint), $\boldsymbol{C} \in \mathbb{R}_+^{N\times T}$ is a temporal matrix that characterizes the fluorescence of each neuron over time, matrix $\boldsymbol{B}$ represents background fluctuations and $\boldsymbol{E}$ is additive Gaussian noise with mean zero and diagonal covariance.

The CNMF framework of [5] incorporates further constraints beyond non-negativity. Each spatial footprint $\boldsymbol{a}_n$ is constrained to be spatially localized and hence sparse. Similarly, the temporal components $\boldsymbol{c}_n$ are highly structured, as they represent the cells' fluorescence responses to typically sparse, nonnegative trains of action potentials. Following [23, 24], we model the calcium dynamics of each neuron $\boldsymbol{c}_n$ with a stable autoregressive process of order P,

$$c_n(t) = \sum_{p=1}^{P} \gamma_p c_n(t-p) + s_n(t), \tag{2}$$

where $s_n(t) \geq 0$ is the number of spikes that neuron n fired at the t-th frame, and γ_p, $p = 1, \ldots, P$ correspond to the discrete time constants of the dynamics that depend on the kinematic properties of the used indicator.

For the case of microendoscopic data we refer the reader to [6] for a very detailed exposition of the model. The background $\boldsymbol{B}$ is modeled as sum of constant baselines $\boldsymbol{B}^c$ and fluctuating activity $\boldsymbol{B}^f$ [6]

$$\boldsymbol{B} = \underbrace{\bar{\boldsymbol{b}}\boldsymbol{1}_{\mathrm{T}}^{\top}}_{\boldsymbol{B}^c} + \underbrace{\boldsymbol{W}(\boldsymbol{Y} - \boldsymbol{AC} - \bar{\boldsymbol{b}}\boldsymbol{1}_{\mathrm{T}}^{\top})}_{\boldsymbol{B}^f}, \tag{3}$$

where $\mathbf{1}_T$ denotes a vector of T ones and the constant baselines are $\bar{\boldsymbol{b}} = \frac{1}{T}(\boldsymbol{Y} - \boldsymbol{AC})\mathbf{1}_T$. The model for $\boldsymbol{B}^f$ exploits that background sources (largely due to blurred out-of-focus fluorescence) are empirically much coarser spatially than the average neuron soma size. Thus we model $\boldsymbol{B}^f$ at one pixel as a linear combination of the background fluorescence in pixels which are chosen to be nearby but not nearest neighbors, $\boldsymbol{B}^f \approx \boldsymbol{W}\boldsymbol{B}^f$. $\boldsymbol{W}$ is an appropriate sparse weight matrix, where W_{ij} is constrained to $W_{ij} = 0$ if $\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) \notin [l, l+1[$, thus we model the background at one pixel as a linear combination of the background fluorescence in pixels which are chosen to be on a ring with radius l [6]. Typically, l is chosen to be $\sim 1.5\times$ the radius of an average neuron, to exclude contributions that might be affected from the activity of an underlying neuron.

Fitting the CNMF-E model

We first recap the offline approach for fitting the CNMF-E model [6], and then show how it can be adapted to an online setup.

Offline. The estimation of all model variables can be formulated as a single optimization problem

$$\underset{\boldsymbol{A},\boldsymbol{C},\boldsymbol{B}}{\text{minimize}} \quad \|\boldsymbol{Y} - \boldsymbol{AC} - \boldsymbol{B}\|_F^2 \quad \text{subject to constraints.} \tag{4}$$

The CNMF-E algorithm of [6] divides the nonconvex problem (4) into three simpler subproblems that are solved iteratively: Estimating $\boldsymbol{A}$ given estimates $\hat{\boldsymbol{C}}$ and $\hat{\boldsymbol{B}}$, estimating $\boldsymbol{C}$ given $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{B}}$, and estimating $\boldsymbol{B}$ given $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{C}}$.

$\boldsymbol{A}$ and $\boldsymbol{C}$ are estimated using a modified version of "fast hierarchical alternating least squares" [25] that includes sparsity and localization constraints [26]. The update of $\boldsymbol{A}$ consists of block-coordinate decent steps iterating over neurons n,

$$\boldsymbol{A}_{\boldsymbol{p}(n),n} \leftarrow \left\lfloor \boldsymbol{A}_{\boldsymbol{p}(n),n} + \frac{((\boldsymbol{Y} - \hat{\boldsymbol{B}})\hat{\boldsymbol{C}}^{\top})_{\boldsymbol{p}(n),n} - (\boldsymbol{A}\hat{\boldsymbol{C}}\hat{\boldsymbol{C}}^{\top})_{\boldsymbol{p}(n),n}}{(\hat{\boldsymbol{C}}\hat{\boldsymbol{C}}^{\top})_{nn}} \right\rfloor_+, \tag{5}$$

where $\boldsymbol{p}(n)$ specifies the pixel indices where $\boldsymbol{A}_{:,n}$ can take non-zero values, i.e. where neuron n is located. For computational efficiency the sufficient statistics $\boldsymbol{L} = (\boldsymbol{Y} - \hat{\boldsymbol{B}})\hat{\boldsymbol{C}}^{\top}$ and $\boldsymbol{M} = \hat{\boldsymbol{C}}\hat{\boldsymbol{C}}^{\top}$ are computed only once initially and cached to be reused when iterating few times over neurons $n \in \{1, \ldots, N\}$.

Similarly, the block-coordinate decent steps for updating $\boldsymbol{C}$ are

$$\boldsymbol{C}_{n,:} \leftarrow \boldsymbol{C}_{n,:} + \frac{(\hat{\boldsymbol{A}}^\top(\boldsymbol{Y} - \hat{\boldsymbol{B}}))_{n,:} - (\hat{\boldsymbol{A}}^\top\hat{\boldsymbol{A}}\boldsymbol{C})_{n,:}}{(\hat{\boldsymbol{A}}^\top\hat{\boldsymbol{A}})_{nn}}, \tag{6}$$

with sufficient statistics $\hat{\boldsymbol{A}}^\top(\boldsymbol{Y} - \hat{\boldsymbol{B}})$ and $\hat{\boldsymbol{A}}^\top\hat{\boldsymbol{A}}$ computed only once initially. $\boldsymbol{C}$ should not merely be constrained to non-negative values but follow the dynamics of the calcium indicator, thus to further denoise and deconvolve the neural activity from the dynamics of the indicator the OASIS algorithm [23] is used. OASIS solves a modified LASSO problem

$$\underset{\boldsymbol{c},\boldsymbol{s}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{c} - \boldsymbol{y}\|^2 + \lambda\|\boldsymbol{s}\|_1 \quad \text{subject to} \quad s_t = c_t - \sum_{p=1}^{P}\gamma_p c_{t-p} \geq s_{\min} \ \text{ or } \ s_t = 0, \tag{7}$$

where $\boldsymbol{y}$ denotes a noisy neural calcium trace obtained as result of Eq (6). The ℓ_1 penalty on $\boldsymbol{s}$ or the minimal spike size s_{min} can be used to enforce sparsity of the neural activity.

The spatiotemporal background is estimated from the linear regression problem

$$\underset{\boldsymbol{W}}{\text{minimize}} \quad \|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{X}\|_F^2 \quad \text{subject to} \quad \boldsymbol{W}_{ij} = 0 \quad \text{if } \operatorname{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) \notin [l, l+1[, \tag{8}$$

where $\boldsymbol{X} = \boldsymbol{Y} - \hat{\boldsymbol{A}}\hat{\boldsymbol{C}} - \bar{\boldsymbol{b}}\boldsymbol{1}_T^\top$ and $\bar{\boldsymbol{b}} = \frac{1}{T}(\boldsymbol{Y} - \hat{\boldsymbol{A}}\hat{\boldsymbol{C}})\boldsymbol{1}_T$. The solution is given by the normal equations for each pixel i,

$$\boldsymbol{W}_{i,\boldsymbol{r}_l(i)} = (\boldsymbol{X}\boldsymbol{X}^\top)_{i,\boldsymbol{r}_l(i)}(\boldsymbol{X}\boldsymbol{X}^\top)^{-1}_{\boldsymbol{r}_l(i),\boldsymbol{r}_l(i)}, \tag{9}$$

where $\boldsymbol{r}_l(i) = \{j|\operatorname{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) \in [l, l+1[\}$ specifies the pixel indices where $\boldsymbol{W}_{i,:}$ can take non-zero values. Given the optimized $\boldsymbol{W}$, the whole background signal is $\boldsymbol{B} = \boldsymbol{W}\boldsymbol{X} + \bar{\boldsymbol{b}}\boldsymbol{1}_T^\top$. More information can be found in [6].

Online. The offline framework presented above can be adapted to a data streaming setup, using the same model assumptions. Instead of running CNMF-E afresh on the entire data seen so far up to time t, $\boldsymbol{Y}[:, 1\!: t]$, the previous estimates $\boldsymbol{A}$, $\boldsymbol{C}$, $\boldsymbol{B}$ obtained on the data up to time $t - 1$, $\boldsymbol{Y}[:, 1\!: t - 1]$, are updated using the newly recorded frame $\boldsymbol{y}_t$, eliminating the need to load the entire data $\boldsymbol{Y}$ in memory and avoiding repetitive computations. One complicating factor is that during online processing some neurons may become active for the first time, thus we need a method to detect those new components and append them to the spatial and temporal matrices $\boldsymbol{A}$ and $\boldsymbol{C}$. In essence, we need to appropriately modify the online algorithm for analyzing 2-photon calcium imaging data (OnACID, [21]) to the case of microendoscopic 1-photon data which requires a more refined background model.

Using Eq (1), the observed fluorescence at time t can be written as

$$\boldsymbol{y}_t = \boldsymbol{A}\boldsymbol{c}_t + \boldsymbol{b}_t + \boldsymbol{\varepsilon}_t. \tag{10}$$

The (non-deconvolved) activity of all neurons at time t, $\boldsymbol{c}_t$, is obtained by iteratively evaluating Eq (6) given raw frame data $\boldsymbol{y}_t$, spatial footprints $\boldsymbol{A}$, and background parameters $\boldsymbol{W}$, $\bar{\boldsymbol{b}}$. The activity is further denoised and deconvolved by running OASIS [23], which is not only a very fast algorithm, but crucially progresses through each time series sequentially from beginning to end and is thus directly applicable to stream processing. Using the expression of $\boldsymbol{b}_t$ (the t-th column of $\boldsymbol{B}$) from Eq (3), the background term in Eq (6) evaluates to $\boldsymbol{A}^\top\boldsymbol{b}_t = \boldsymbol{A}^\top\boldsymbol{W}\boldsymbol{y}_t - \boldsymbol{A}^\top\boldsymbol{W}\boldsymbol{A}\boldsymbol{c}_t - \boldsymbol{A}^\top\boldsymbol{W}\bar{\boldsymbol{b}} + \boldsymbol{A}^\top\bar{\boldsymbol{b}}$ and for computational efficiency the terms $\boldsymbol{A}^\top\ \boldsymbol{W}$, $\boldsymbol{A}^\top\boldsymbol{W}\boldsymbol{A}$ and $\boldsymbol{A}^\top(\boldsymbol{W}\bar{\boldsymbol{b}} - \bar{\boldsymbol{b}})$ are maintained in memory and updated incrementally, cf. S1 Appendix. Warm starts are exploited by initializing $\boldsymbol{c}_t$ with the value at the previous frame $\boldsymbol{c}_{t-1}$, since the calcium traces $\boldsymbol{C}$ are continuous and typically change slowly. Moreover, the temporal traces of components that do not spatially overlap with each other can be updated simultaneously in vector form; we use a simple greedy scheme to partition the components into spatially non-overlapping groups [21].

The spatial footprints $\boldsymbol{A}$ are obtained by iteratively evaluating Eq (5) and can be estimated efficiently as in [22] by only keeping in memory the sufficient statistics

$$\boldsymbol{L}_t = \frac{t-1}{t}\boldsymbol{L}_{t-1} + \frac{1}{t}(\boldsymbol{y}_t - \boldsymbol{b}_t)\boldsymbol{c}_t^\top, \qquad \boldsymbol{M}_t = \frac{t-1}{t}\boldsymbol{M}_{t-1} + \frac{1}{t}\boldsymbol{c}_t\boldsymbol{c}_t^\top. \tag{11}$$

Since neurons' shapes are not expected to change at a fast timescale, updating $\boldsymbol{A}$ is actually not required at every timepoint; in practice we update every 200 time steps, again warm started at the value from the previous iteration, cf. Alg 1. Additionally, the sufficient statistics $\boldsymbol{L}_t$, $\boldsymbol{M}_t$ are only needed for updating the estimates of $\boldsymbol{A}$ so they can be updated only when required (using computationally efficient matrix products). Further, only a sparse subset of the elements of matrix $\boldsymbol{L}$ needs to be updated, because Eq (5) does not access all elements of $\boldsymbol{L}$, but only elements $\boldsymbol{p}(n)$ for each column n of $\boldsymbol{L}$ (i.e. only pixel indices $\boldsymbol{p}(n)$ where neuron n is located). Hence, we speed up the algorithm by updating only those few entries of $\boldsymbol{L}$, cf. S1 Appendix.

To update the background components $\boldsymbol{W}$, $\bar{\boldsymbol{b}}$, we keep track of the constant baselines $\bar{\boldsymbol{b}}$ and the sufficient statistics $\boldsymbol{\chi} = \boldsymbol{X}\boldsymbol{X}^\top$ that is needed to compute $\boldsymbol{W}$ using Eq (9)

$$\bar{\boldsymbol{b}}_t \leftarrow \frac{t-1}{t}\bar{\boldsymbol{b}}_{t-1} + \frac{1}{t}(\boldsymbol{y}_t - \boldsymbol{A}\boldsymbol{c}_t), \qquad \boldsymbol{\chi}_t = \frac{t-1}{t}\boldsymbol{\chi}_{t-1} + \frac{1}{t}\boldsymbol{x}_t\boldsymbol{x}_t^\top, \tag{12}$$

where $\boldsymbol{x}_t = \boldsymbol{y}_t - \boldsymbol{A}\boldsymbol{c}_t - \bar{\boldsymbol{b}}_t$. As is the case with the spatial footprints, updating the background is actually not required at every timepoint and in practice we update every 200 time steps, cf. Alg 1 and S1 Appendix. Processing pixel i according to Eq (9) (see also S1 Appendix) accesses only vector $\boldsymbol{\chi}_{i,\boldsymbol{r}_l(i)}$ and sub-matrix $\boldsymbol{\chi}_{\boldsymbol{r}_l(i),\boldsymbol{r}_l(i)}$. Some elements of $\boldsymbol{\chi}$ are not part of any sub-matrix or vector for any i and thus are never accessed. In practice we therefore update and store only these vectors and sub-matrices for computational and memory efficiency. Because the background has no high spatial frequency components, it can be spatially decimated to further speed up processing [23] without compromising the quality of the results. E.g. downscaling by a factor of 2 reduces the number of pixels by a factor of 4 and the number of elements in $\boldsymbol{W}$ and $\boldsymbol{\chi}$ by a factor of 16. Less and smaller least squares problems (Eq 9) need to be solved, which drastically reduces processing time and memory consumption.

Note that updating the background components and all the spatial footprints at a given frame results in a computational bottleneck for that specific frame. While on average, this effect is minimal (cf. Results section) a temporary slowdown can have an adverse effect on a real-time closed loop setup. This restriction can be lifted by holding the background model fixed and updating the spatial footprints in a distributed manner across all frames. As described later, using a lower dimensional background model can achieve that and enable fast real time processing with balanced workload across all frames.

To initialize our algorithm we use the CNMF-E algorithm on a short initial batch of data of length T_b, (e.g., $T_b = 200$). The sufficient statistics are initialized from the components that the offline algorithm finds according to Eqs (11) and (12).

Algorithm 1 OnACID-E

```
Require: Data matrix $\boldsymbol{Y}$, initial estimates $\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{S}, \boldsymbol{W}, \bar{\boldsymbol{b}}$, current number of
components $N$, current time step $t'$, rest of parameters.
1: $\boldsymbol{X} = \boldsymbol{Y}[:, 1:t'] - \boldsymbol{A}\boldsymbol{C} - \bar{\boldsymbol{b}}1_{t'}^\top$
2: $\boldsymbol{R}_{\text{buf}} = (\boldsymbol{X} - \boldsymbol{W}\boldsymbol{X})[:, t' - l_b + 1 : t']$        ▷ Initialize residual buffer
3: $\boldsymbol{\chi} = \boldsymbol{X}\boldsymbol{X}^\top$                                  ▷ Initialize sufficient statistics
4: $\boldsymbol{L} = \boldsymbol{Y}[:, 1:t']\boldsymbol{C}^\top/t'$
```

```
5: $\boldsymbol{M} = \boldsymbol{C}\boldsymbol{C}^\top / t'$
6: $\mathcal{G} = \textsc{DetermineGroups}(\boldsymbol{A}, N)$    ▷ [21]
7: $t = t'$
8: while there is more data do
9:   $t \leftarrow t + 1$
10:    $\boldsymbol{y}_t \leftarrow \textsc{AlignFrame}(\boldsymbol{y}_t, \boldsymbol{A}\boldsymbol{c}_{t-1})$    ▷ Alg S6
11:    $\boldsymbol{c}_t \leftarrow \textsc{UpdateTraces}(\boldsymbol{A}, \boldsymbol{c}_{t-1}, \boldsymbol{y}_t, \boldsymbol{W}, \bar{\boldsymbol{b}}, \mathcal{G})$    ▷ Alg S1
12:    $\boldsymbol{C}, \boldsymbol{S} \leftarrow \text{OASIS}(\boldsymbol{C}, \boldsymbol{\gamma}, \boldsymbol{s}_{\min}, \lambda)$    ▷ [23]
13:    $\bar{\boldsymbol{b}} \leftarrow \frac{t-1}{t}\bar{\boldsymbol{b}} + \frac{1}{t}(\boldsymbol{y}_t - \boldsymbol{A}\boldsymbol{c}_t)$
14:    $\boldsymbol{A}, \boldsymbol{C}, N, \mathcal{G}, \boldsymbol{R}_{\text{buf}} \leftarrow \textsc{DetectNewComponents}(\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{W}, \bar{\boldsymbol{b}}, N, \mathcal{G}, \boldsymbol{R}_{\text{buf}}, \boldsymbol{y}_t)$    ▷ Alg S5
15:    if $\mathrm{mod}(t - t', T_p) = 0$ then    ▷ Update $\boldsymbol{\chi}$, $\boldsymbol{L}$, $\boldsymbol{M}$, $\boldsymbol{W}$, $\boldsymbol{A}$ every $T_p$ time steps
16:      $\boldsymbol{\chi}, \boldsymbol{L}, \boldsymbol{M} \leftarrow \textsc{UpdateSuffStatistics}(\boldsymbol{Y}[:, t - T_p{+}1:t], \boldsymbol{C}[:, t - T_p{+}1:t], \boldsymbol{W}, \bar{\boldsymbol{b}}, \boldsymbol{A}, \boldsymbol{\chi}, \boldsymbol{L}, \boldsymbol{M})$    ▷ Alg S2
17:      $\boldsymbol{W} \leftarrow \textsc{UpdateBackground}(\boldsymbol{\chi})$    ▷ Alg S4
18:      $\boldsymbol{A} \leftarrow \textsc{UpdateShapes}(\boldsymbol{L}, \boldsymbol{M}, \boldsymbol{A})$    ▷ Alg S3
19: return $\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{S}, \boldsymbol{W}, \bar{\boldsymbol{b}}$
```

Detecting new components. The approach explained above enables tracking the activity of a fixed number of sources, and will ignore neurons that become active later in the experiment. Following [21], we approach the problem by introducing a buffer $\boldsymbol{R}_{\text{buf}}$ that contains the last l_b instances of the residual signal $\boldsymbol{r}_t = \boldsymbol{y}_t - \boldsymbol{A}\boldsymbol{c}_t - \boldsymbol{b}_t$ ($\boldsymbol{R}_{\text{buf}} = [\boldsymbol{r}_{t-l_b+1}, \ldots \boldsymbol{r}_t]$), where l_b is a reasonably small number, e.g., $l_b = 100$. From this buffer we compute a summary image (as detailed later we actually update the summary image instead of computing it afresh) and then search for the local maxima of the image to determine new candidate neurons.

One option for the summary image $\boldsymbol{e}$ is to proceed along the lines of [5], i.e. to perform spatial smoothing with a Gaussian kernel with radius similar to the expected neuron radius, and then calculate the energy for each pixel i, $\boldsymbol{e}[i] = \frac{1}{l_b}\sum_t \text{filt}(\boldsymbol{R}_{\text{buf}}[i,t])^2$, where filt() refers to the smoothing operation. Another option is to follow [6] and calculate the peak-to-noise ratio (PNR),

$$\boldsymbol{i}_{\text{pnr}}[i] = \frac{\max_t \boldsymbol{R}_{\text{buf}}[i,t]}{\sigma_i}, \tag{13}$$

as well as the local cross-correlation image,

$$\boldsymbol{i}_{\text{corr}}[i] = \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} \text{corr}(\boldsymbol{R}_{\text{buf}}[i,:], \boldsymbol{R}_{\text{buf}}[j,:]), \tag{14}$$

where $\mathcal{N}(i)$ specifies the neighboring pixels of pixel i and the function corr() refers to Pearson correlation. Their pixel-wise product $\boldsymbol{e} = \boldsymbol{i}_{\text{pnr}} \odot \boldsymbol{i}_{\text{corr}}$ is used as summary image. We use the latter throughout the Results section, if not explicitly stated otherwise. New candidate components $\boldsymbol{a}_{\text{new}}$, and $\boldsymbol{c}_{\text{new}}$ are estimated by performing a local rank-1 NMF of the residual matrix restricted to a fixed neighborhood around the point of maximal variance, or maximal product of PNR and cross-correlation, respectively.

To limit false positives, the candidate component is screened for quality. Similarly to [21], to prevent noise overfitting, the shape $\boldsymbol{a}_{\text{new}}$ must be significantly correlated (e.g., $\theta_{\text{sp}} \sim 0.5$) to the residual buffer averaged over time and restricted to the spatial extent of $\boldsymbol{a}_{\text{new}}$. Moreover, if $\boldsymbol{a}_{\text{new}}$ significantly overlaps with any of the existing components, then its temporal component $\boldsymbol{c}_{\text{new}}$ must not be highly correlated with the corresponding temporal components; otherwise we reject it as a possible duplicate of an existing component. Further, for a candidate component to correspond to an active neuron its trace must exhibit dynamics reminiscent of the calcium indicator's transient. As criterion for this we require the SNR of trace $\boldsymbol{c}_{\text{new}}$ to be above a certain threshold θ_{SNR}. We tuned the spatial and temporal thresholds for each considered dataset. Once a new component is accepted, $\boldsymbol{A}$, $\boldsymbol{C}$ are augmented with $\boldsymbol{a}_{\text{new}}$ and $\boldsymbol{c}_{\text{new}}$ respectively, the quantities $\boldsymbol{A}^\top\ \boldsymbol{W}$, $\boldsymbol{A}^\top\ \boldsymbol{WA}$ and $\boldsymbol{A}^\top(\boldsymbol{W}\bar{\boldsymbol{b}} - \bar{\boldsymbol{b}})$ are updated via augmentation, and the sufficient statistics are updated as follows:

$$\boldsymbol{L}_t = \left[\boldsymbol{L}_t, \frac{1}{t}(\boldsymbol{Y}_{\text{buf}} - \boldsymbol{B}_{\text{buf}})\boldsymbol{c}_{\text{new}}^\top\right], \qquad \boldsymbol{M}_t = \frac{1}{t}\begin{bmatrix} t\boldsymbol{M}_t & \boldsymbol{C}_{\text{buf}}\boldsymbol{c}_{\text{new}}^\top \\ \boldsymbol{c}_{\text{new}}\boldsymbol{C}_{\text{buf}}^\top & \|\boldsymbol{c}_{\text{new}}\|^2 \end{bmatrix}, \tag{15}$$

where $\boldsymbol{Y}_{\text{buf}}, \boldsymbol{C}_{\text{buf}}, \boldsymbol{B}_{\text{buf}} = \bar{\boldsymbol{b}}\boldsymbol{1}_{l_b}^\top + \boldsymbol{W}(\boldsymbol{Y}_{\text{buf}} - \boldsymbol{AC}_{\text{buf}} - \bar{\boldsymbol{b}}\boldsymbol{1}_{l_b}^\top)$ denote the matrices $\boldsymbol{Y}$, $\boldsymbol{C}$, $\boldsymbol{B}$, restricted to the last l_b frames that the buffer stores. This process is repeated until all candidates have been screened, at which point the next frame is read and processed. The process is summarized in Alg S5.

Updating the summary image. For computational efficiency we avoid repeated computations and perform incremental updates of the summary image instead of computing it afresh. If the variance image is used, it is updated according to $\boldsymbol{e} \leftarrow \boldsymbol{e} + \frac{1}{l_b}(\text{ filt }(\boldsymbol{r}_t)^2 - \text{ filt }(\boldsymbol{r}_{t-l_b})^2)$ when the next frame is processed. When a new component with footprint $\boldsymbol{a}$ is added the residual changes at the component's location and we update the variance image accordingly locally only for pixels i where the smoothed component is positive ($\text{filt}(\boldsymbol{a})[i] > 0$) according to $\boldsymbol{e}[i] \leftarrow \frac{1}{l_b}\sum_t \text{filt }(\boldsymbol{R}_{\text{buf}}[i, t])^2$.

Next we consider the case that the product of cross-correlation image and PNR image is used as summary image. We keep track of the first and second order statistics

$$\mu_i = \frac{1}{l_b}\sum_t \boldsymbol{R}_{\text{buf}}[i, t] \quad \text{and} \quad \nu_{ij} = \frac{1}{l_b}\sum_t \boldsymbol{R}_{\text{buf}}[i, t]\boldsymbol{R}_{\text{buf}}[j, t], \tag{16}$$

the latter only for pixels $j \in \{i\} \cup \mathcal{N}(i)$. These statistics are updated according to

$$\boldsymbol{\mu} \quad \leftarrow \boldsymbol{\mu} + \tfrac{1}{l_b}(\boldsymbol{r}_t - \boldsymbol{r}_{t-l_b}) \tag{17}$$

$$\nu_{ij} \quad \leftarrow \nu_{ij} + \tfrac{1}{l_b}(\boldsymbol{r}_t\boldsymbol{r}_t^\top - \boldsymbol{r}_{t-l_b}\boldsymbol{r}_{t-l_b}^\top)_{ij} \tag{18}$$

when the next frame is processed. The cross-correlation values are computed from these statistics as

$$\text{corr}(\boldsymbol{R}_{\text{buf}}[i, :], \boldsymbol{R}_{\text{buf}}[j, :]) = \frac{\nu_{ij} - \mu_i\mu_j}{\sqrt{(\nu_{ii} - \mu_i^2)(\nu_{jj} - \mu_j^2)}}, \tag{19}$$

and the correlation image is obtained according to Eq (14). For computing the PNR image we use the noise level σ_i estimated on the small initial batch for the denominator in Eq (13) and keep track of the maximum image $\boldsymbol{i}_{\max} \leftarrow \max(\boldsymbol{i}_{\max}, \boldsymbol{r}_t)$ for the nominator. When a new component with footprint $\boldsymbol{a}$ and time series $\tilde{\boldsymbol{c}}$ is added we set $\boldsymbol{i}_{\max}[i]$ to zeros if $a_i > 0$. The statistics for the cross-correlation are updated as

$$\mu_i \quad \leftarrow \mu_i - \frac{1}{l_b}\sum\nolimits_t \tilde{c}_t a_i \tag{20}$$

$$v_{ij} \quad \leftarrow v_{ij} + \frac{1}{l_b}\sum\nolimits_t (\tilde{c}_t^2 a_i a_j - \boldsymbol{R}_{\text{buf}}[j,t]\tilde{c}_t a_i - \boldsymbol{R}_{\text{buf}}[i,t]\tilde{c}_t a_j). \tag{21}$$

The whole online procedure of OnACID-E is described in Algorithm 1; S1 Appendix includes pseudocode description of the referenced routines.

Background modeling using convolutional neural networks

The background model used in the CNMF-E algorithm (Eq 8) assumes that the value of the background signal at a given point in space is given by a linear combination of the background values from the points in a ring centered around that pixel with width 1 and radius *l*, where *l* is larger than the radius of the typical neuron in the dataset by a small factor (e.g. 1.5) plus a pixel dependent scalar [6]. While powerful in practice, this model does not assume any dependence between the linear combination weights of all the different pixels, and results in a model with a very large number of parameters to be estimated. Ignoring pixels near the boundary, each row of the matrix $\boldsymbol{W}$ which represents the linear combination weights will have approximately $[2\pi l]$ non-zero entries (where $[\cdot]$ denotes the integer part), giving a total number of $d([2\pi l] + 1)$ parameters to be estimated. While this estimation can be done efficiently in parallel as discussed above, and the overall number of parameters can be reduced through spatial downsampling, we expect that the overall number of degrees of freedom in such a model is much lower. The reason is that the ring model of CNMF-E aims to capture aspects of the point spread function (PSF) which is largely invariant with respect to the location within the local environment of each neuron.

To test this hypothesis we used a very simple convolutional neural network (CNN) with ring shaped kernels to capture the background structure. The intuition behind the convolution is straightforward: if all the rows of the $\boldsymbol{W}$ matrix had the same non-zero entries (but centered around different points) then the application of $\boldsymbol{W}$ would correspond to a simple spatial convolution with the common "ring" as the filter. In our case this model is not expressive enough to adequately fit the background, in particular it fails to capture pixel dependent brightness differences, and by assuming shift invariance it fails to capture that the PSF can vary when compared across the full FOV. Therefore we investigated parametrizing the background model with a slightly more complex model, which we refer to as "Ring-CNN".

Let $f_{\boldsymbol{\theta}} : \mathbb{R}^d \mapsto \mathbb{R}^d$ be a function that models the autoregressive nature of the background. In the CNMF-E case this simply corresponds to $f_{\boldsymbol{\theta}}(\boldsymbol{y} - \boldsymbol{Ac}) = \boldsymbol{W}(\boldsymbol{y} - \boldsymbol{Ac} - \bar{\boldsymbol{b}}) + \bar{\boldsymbol{b}}$, cf. Eq (3). In the linear model we parametrize the function as

$$f_{\boldsymbol{\theta}}(\boldsymbol{y}) = \sum_{k=1}^{K} \boldsymbol{w}_k \odot (\boldsymbol{h}_k * \boldsymbol{y}) + \bar{\boldsymbol{b}}, \tag{22}$$

where $\bar{\boldsymbol{b}}, \boldsymbol{w}_k \in \mathbb{R}^d, k = 1, \ldots, K$, and $\odot, *$ refer to pointwise multiplication and spatial convolution, respectively (with slight abuse of notation we assume that $\boldsymbol{y}$ has been reshaped back to 2d image to perform the convolution and the result of the convolution is again vectorized). Finally, $\boldsymbol{h}_k$, $k = 1, \ldots, K$ is a ring shaped convolutional kernel which takes non-zero values only at a specified annulus around its center. Note that this corresponds to parametrizing directly

$\boldsymbol{W}$ as

$$\boldsymbol{W} = \sum_{k=1}^{K} \boldsymbol{w}_k \odot \boldsymbol{H}_k, \tag{23}$$

where $\boldsymbol{H}_k \in \mathbb{R}^{d \times d}$ is the matrix induced by the convolutional kernel $\boldsymbol{h}_k$. Constructing the sparse matrix $\boldsymbol{W}$ explicitly (and efficiently using diagonal storage) can speed up evaluation of the background when a GPU is not available. Intuitively this model corresponds to using a pixel dependent linear combination of K ring basis functions, and results in a total $K(d + [2\pi l]) + d$ parameters to be estimated. Compared to the $d([2\pi l] + 1)$ number of parameters for the CNMF-E model, this can result in a significant reduction when $K < [2\pi l]$.

Note that decoupling the number of different "rings" from the total number of pixels, enables the consideration of wider "rings" that integrate over a larger area of the FOV and can potentially provide more accurate estimates, without a dramatic increase on the number of parameters to be learned. For example, a "ring" with inner radius l and width w would require approximately $[\pi w(2l + w - 1)]$ parameters and the total number of parameters would be $K([\pi w(2l + w - 1)] + d)$ as opposed to $d([\pi w(2l + w - 1)] + 1)$ for the standard CNMF-E model.

Unsupervised training on the raw data. To estimate the autoregressive background model in the CNMF-E algorithm, we want to operate on the data after the spatiotemporal activity of all detected neurons has been removed (Eq 8). For the CNN model this would translate into the optimization problem

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{Y} - \boldsymbol{AC}, f_{\boldsymbol{\theta}}(\boldsymbol{Y} - \boldsymbol{AC})), \tag{24}$$

where $\mathcal{L}(\cdot,\cdot) : \mathbb{R}^{d \times T} \times \mathbb{R}^{d \times T} \mapsto \mathbb{R}_+$ is an appropriate loss function (e.g. the Frobenius norm).

For the CNMF-E algorithm, operating on $\boldsymbol{Y} - \boldsymbol{AC}$ is necessary because each "ring" has its own independent weights whose estimation can be biased from the activity of nearby neurons. In the CNN case however, the background model assumes a significant amount of weight sharing between the different "rings" which makes the estimation more robust to the underlying neural activity. Therefore we can estimate the background model by solving directly

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{Y}, f_{\boldsymbol{\theta}}(\boldsymbol{Y})), \tag{25}$$

for an appropriately choosen loss function $\mathcal{L}$ meaning that the solution of Eq (25) should satisfy $f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{Y}) \approx \boldsymbol{Y} - \boldsymbol{AC}$. Because $\boldsymbol{Y} - \boldsymbol{AC} \approx f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{Y} - \boldsymbol{AC}) = f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{Y}) - f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{AC})$, the underlying assumption is that $f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{AC})$ can be neglected. While the autoregressive model can capture the background well, i.e. $\boldsymbol{Y} - \boldsymbol{AC} \approx f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{Y} - \boldsymbol{AC})$, this is not the case for the neural activity, because the fluorescence $\boldsymbol{AC}$ at each pixel due to neural activity can not be reconstructed well using the unrelated fluorescence traces on the ring around that pixel $f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{AC})$, i.e. $\boldsymbol{AC} \not\approx f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{AC})$ independent of $\hat{\boldsymbol{\theta}}$. This particularly holds during training of the CNN where not just one but hundreds of frames are reconstructed, ruling out overfitting. Thus including the neural activity $\boldsymbol{AC}$ in the objective, Eq (25) instead of Eq (24), hardly affects the optimal parameters $\hat{\boldsymbol{\theta}}$. Furthermore, since $\boldsymbol{AC}$ is nonnegative we seek to under-approximate $\boldsymbol{Y}$ with the background $f_{\boldsymbol{\theta}}(\boldsymbol{Y})$. To encode that in the objective function we can consider a quantile loss function [27] that penalizes over-approximation more than under-approximation:

$$l_q(x,y) = \begin{cases} q(x-y), & x \geq y \\ (1-q)(y-x), & x < y \end{cases}, \tag{26}$$

For some $q \in (0, 1]$ and take $\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}) = 2\sum_{i,j} l_q(\boldsymbol{X}_{ij}, \boldsymbol{Y}_{ij})$. For example, for $q = 0.5$ Eq (26) corresponds to the L_1 norm of the difference, and to promote the under-approximation property we use $q < 0.5$. Since the models are differentiable and the objective function is additive, Eq (25) can be optimized in an online mode using stochastic gradient descent.

Online processing.

Algorithm 2 Online processing with a Ring-CNN background model

```
Require: Data matrix $\boldsymbol{Y}$, number of initial timesteps $T_{\text{init}}$, rest of parameters.
1: $\boldsymbol{X}$ = MotionCorrect($\boldsymbol{Y}$[:, 1 : $T_{\text{init}}$])    ▷ [28]
2: $\hat{\boldsymbol{\theta}} \leftarrow \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{X}, f_{\boldsymbol{\theta}}(\boldsymbol{X}))$    ▷ Estimate ring CNN (25)
3: $\boldsymbol{X} \leftarrow \boldsymbol{X} - \boldsymbol{f}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{X})$    ▷ Filter Background
4: $\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{S}, \boldsymbol{b}, \boldsymbol{f}$, = InitializeOnline2P($\boldsymbol{X}$)    ▷ Initialize online algorithm [17]
5: $t = T_{\text{init}}$
6: while there is more data do
7:     $t \leftarrow t + 1$
8:     $\boldsymbol{y}_t \leftarrow$ AlignFrame($\boldsymbol{y}_t$, $\boldsymbol{b}_{t-1} + \boldsymbol{A}\boldsymbol{c}_{t-1}$)    ▷ Alg S6
9:     $\boldsymbol{x}_t = \boldsymbol{y}_t - \boldsymbol{f}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{y}_t - \boldsymbol{A}\boldsymbol{c}_{t-1})$    ▷ Remove background from current frame
10:    [$\boldsymbol{c}_t$; $\boldsymbol{f}_t$] $\leftarrow$ UpdateTraces2P($\boldsymbol{A}$, [$\boldsymbol{c}_{t-1}$; $\boldsymbol{f}_{t-1}$], $\boldsymbol{x}_t$, $\boldsymbol{b}$, $\boldsymbol{f}$)    ▷ [21, Alg S3]
11:    $\boldsymbol{C}, \boldsymbol{S} \leftarrow$ OASIS($\boldsymbol{C}$, $\gamma$, $\boldsymbol{s}_{\min}$, $\lambda$)    ▷ [23]
12:    $\boldsymbol{A}, \boldsymbol{C}, N, \boldsymbol{R}_{\text{buf}} \leftarrow$ DetectNewComponents2P($\boldsymbol{A}$, $\boldsymbol{C}$, $\boldsymbol{R}_{\text{buf}}$, $\boldsymbol{x}_t$)    ▷ [21, Alg S4]
13:    [$\boldsymbol{A}$, $\boldsymbol{b}$] $\leftarrow$ UpdateShapes2P($\boldsymbol{L}$, $\boldsymbol{M}$, [$\boldsymbol{A}$, $\boldsymbol{b}$])    ▷ [21, Alg S5]
14:    if mod ($t - T_{\text{init}}$, $T_p$) = 0 then    ▷ Update $\boldsymbol{L}, \boldsymbol{M}$ every $T_p$ time steps
15:        $\boldsymbol{L}, \boldsymbol{M} \leftarrow$ UpdateSuffStatistics2P($\boldsymbol{Y}$, $\boldsymbol{C}$, $\boldsymbol{A}$, $\boldsymbol{L}$, $\boldsymbol{M}$)    ▷ [21]
16: return $\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{S}, \boldsymbol{b}, \boldsymbol{f}, \hat{\boldsymbol{\theta}}$
```

In practice, we found that by using rings of increased width (e.g. 5 pixels), training the model only during the initialization process on a small batch frames, leads to convergence due to the large amount of weight sharing that reduces the number of parameters. Once the model has been trained, it can be used to remove the background from the data (after motion correction). To reduce the effect of active neurons on the inferred background we can approximate the activity at time *t*, with the activity at time *t* – 1, and subtract that from the data frame prior to computing the background. In other words, we can use the approximation

$$\boldsymbol{b}_t \simeq f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{y}_t - \boldsymbol{A}\boldsymbol{c}_{t-1}). \tag{27}$$

Once the background has been removed, online processing can be done using the standard online algorithm for two-photon data [21]. The process is summarized in Alg 2, where the suffix "2P" has been added to some routines to indicate their differences compared to the routines used in OnACID-E that are slightly more complicated due to their additional background treatment step. Note that although the focus of this paper is on online processing, the ring-CNN background model can also be used to derive an offline algorithm for microendoscopic 1p data.

Online motion correction

Similarly to [21], online motion correction can be achieved by using the previously denoised frame $\boldsymbol{b}_{t-1} + \boldsymbol{A}\boldsymbol{c}_{t-1}$ to derive a template for registering $\boldsymbol{y}_t$. In practice, we observed that this registration process is more robust to drift introduced by corrupt frames when an average of the past M denoised frames is used as a template, with $M \sim 50$. As proposed in [17], passing both the template and the frame through a high pass spatial filter can suppress the strong background signal present in microendoscopic 1-photon data, and lead to more accurate computation of the alignment transformation. Rigid or piecewise rigid translations can be estimated as described in [28]. The inferred transformation is then applied to original frame $\boldsymbol{y}_t$. The process is summarized in Alg S6.

Analysis details

The detection of components in CNMF-E is controlled by thresholds on the minimum peak-to-noise ratio of seed pixels, P_{min}, and on the minimum local correlation of seed pixels, L_{min}. OnACID-E imposes a threshold $\boldsymbol{\theta}_{\text{sp}}$ on the correlation between a component's spatial footprint and the data averaged over time, as well as a threshold $\boldsymbol{\theta}_{\text{SNR}}$ on the signal to noise ratio. These parameters should be adjusted for the considered dataset and are listed in Table 1. P_{min} and L_{min} can be adjusted based on inspection of the PNR and cross-correlation summary images over the initial batch to account for different noise levels between data sets.

To compare the results between two algorithms we registered the components using the method implemented in CaImAn (with default parameters) and described in [17]. It constructs a matrix of pairwise distances between two components and computes an optimal matching between the components using the Hungarian algorithm to solve the linear assignment problem. Components detected by both algorithms are true positives (TP). Components additionally detected by the first algorithm are false negatives (FN), and false positives (FP) for the second. The accuracy of the agreement is measured by the F1-Score, which is defined as the harmonic mean of the precision and recall, and is in terms of type I and type II errors given by

$$F_1 = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}. \tag{28}$$

The output of the ring CNN is invariant to rotations of its parameters: Assembling the (vectorized) kernels into a matrix $\boldsymbol{H} \coloneqq [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_K]$ we obtain new kernels $\tilde{\boldsymbol{H}}$ by multiplying with a rotation matrix $\boldsymbol{R}$, $\tilde{\boldsymbol{H}} = \boldsymbol{HR}$. Likewise, assembling the weights into a matrix $\boldsymbol{W} \coloneqq [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K]$ we obtain new weights $\tilde{\boldsymbol{W}} = \boldsymbol{WR}$. The rotation leaves the matrix product $\tilde{\boldsymbol{W}}\tilde{\boldsymbol{H}}^\top = \boldsymbol{WR}(\boldsymbol{HR})^\top = \boldsymbol{WRR}^\top\boldsymbol{H}^\top = \boldsymbol{WH}^\top$ invariant, and thus does not change the output of the CNN. To obtain convolution kernels that are ordered by 'importance' we can perform a singular value decomposition (SVD) of $\boldsymbol{W}\,\boldsymbol{H}^\top = \boldsymbol{USV}^\top$, where the singular values in the diagonal matrix $\mathbf{S}$ are ordered form largest to smallest. We set $\tilde{\boldsymbol{W}} = \boldsymbol{US}_{\text{u}}$, $\tilde{\boldsymbol{H}}^\top = \boldsymbol{S}_{\text{v}}\boldsymbol{V}^\top$, with diagonal scaling matrices $\boldsymbol{S}_{\text{u}}$, $\boldsymbol{S}_{\text{v}}$ such that $\boldsymbol{S} = \boldsymbol{S}_{\text{u}}\,\boldsymbol{S}_{\text{v}}$ and depict those in the Results section. Note that the SVD approach would also allow to post-select the number of convolution kernels K. One

Table 1. Thresholds controlling the detection of components for the analyzed datasets.

Dataset	Striatum	PFC	Hippocampus	BNST
L_{min}	0.7	0.9	0.9	0.92
P_{min}	7	15	15	15
θ_{sp}	0.55	0.9	0.85	0.6
θ_{SNR}	3.5	2.8	2.8	3.5

https://doi.org/10.1371/journal.pcbi.1008565.t001 could start with an upper estimate of K, train the CNN, perform SVD of $\boldsymbol{W}\boldsymbol{H}^\top$, and only keep the singular vectors for which the singular value is above some threshold.

Results

Online analysis of 1p microendoscopic data using OnACID-E

We tested the online CNMF-E implementation of OnACID-E on in vivo microendosopic data from mouse dorsal striatum, with neurons expressing GCaMP6f. The data was acquired while the mouse was freely moving in an open field arena. The dataset consisted of 6000 frames at 10 Hz resolution (for further details refer to [6, 29]). We initialized the online algorithm by running CNMF-E on the first 200 frames.

We illustrate OnACID-E in process in Fig 1. At the beginning of the experiment (Fig 1 left), only some components are active, as shown in panel A by the correlation image computed using the spatially filtered data [6], and most of these are detected by the algorithm (N = 218), while avoiding false positives. As the experiment proceeds more neurons activate and are subsequently detected by OnACID-E (Fig 1 middle, N = 337, and right, N = 554), which also tracks their activity across time (Fig 1B). See also S1 Video for further illustration.

Comparison of OnACID-E with CNMF-E

In Fig 2 we report the results of the analysis using OnACID-E and compare to the results of CNMF-E with patches, i.e. the field of view (FOV) is split into smaller overlapping patches that are processed in parallel and combined at the end [17]. For each algorithm, after the processing was done, the identified components were merged, and then screened for false positive using the tests employed in the CaImAn package. Both implementations detect similar components (Fig 2A) with an F1-score of 0.891 (0.875 if the variance summary image was used to detect new components). 506 components were found in common by both implementations. 48 and 76 additional components were detected by OnACID-E and CNMF-E respectively. The additional components are depicted in S1 and S2 Figs respectively, and most of them appear to be

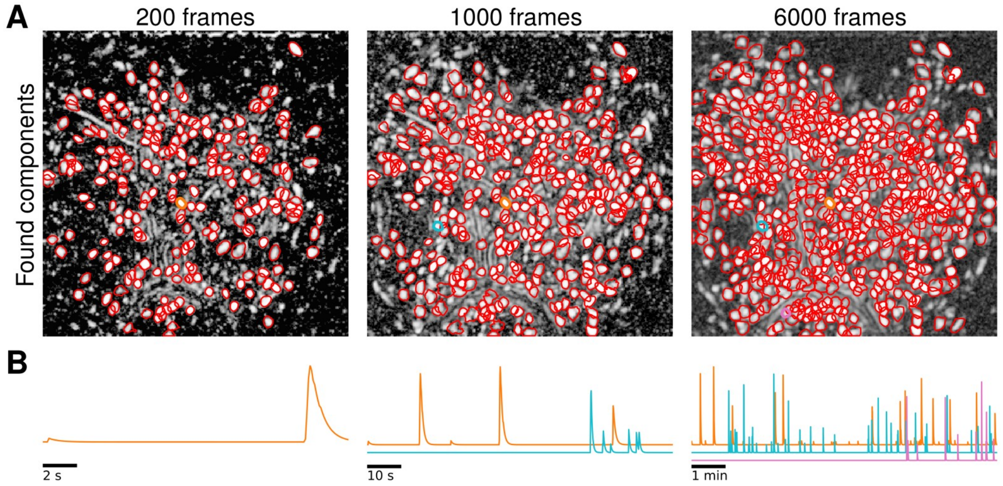

Fig 1. Illustration of the online data analysis process. Snapshots of the online analysis after processing 200 frames (left), 1000 frames (middle), and 6000 frames (right). **(A)** Contours of the components (neurons and processes) found by OnACID-E up to each snapshot point, overlaid over the local cross-correlation image of the spatially filtered data [6] at that point. **(B)** Examples of neuron activity traces (marked by corresponding colors in panel A). As the experiment proceeds, OnACID-E detects newly active neurons and tracks their activity. A video showing the whole online analysis can be found at S1 Video.

https://doi.org/10.1371/journal.pcbi.1008565.g001

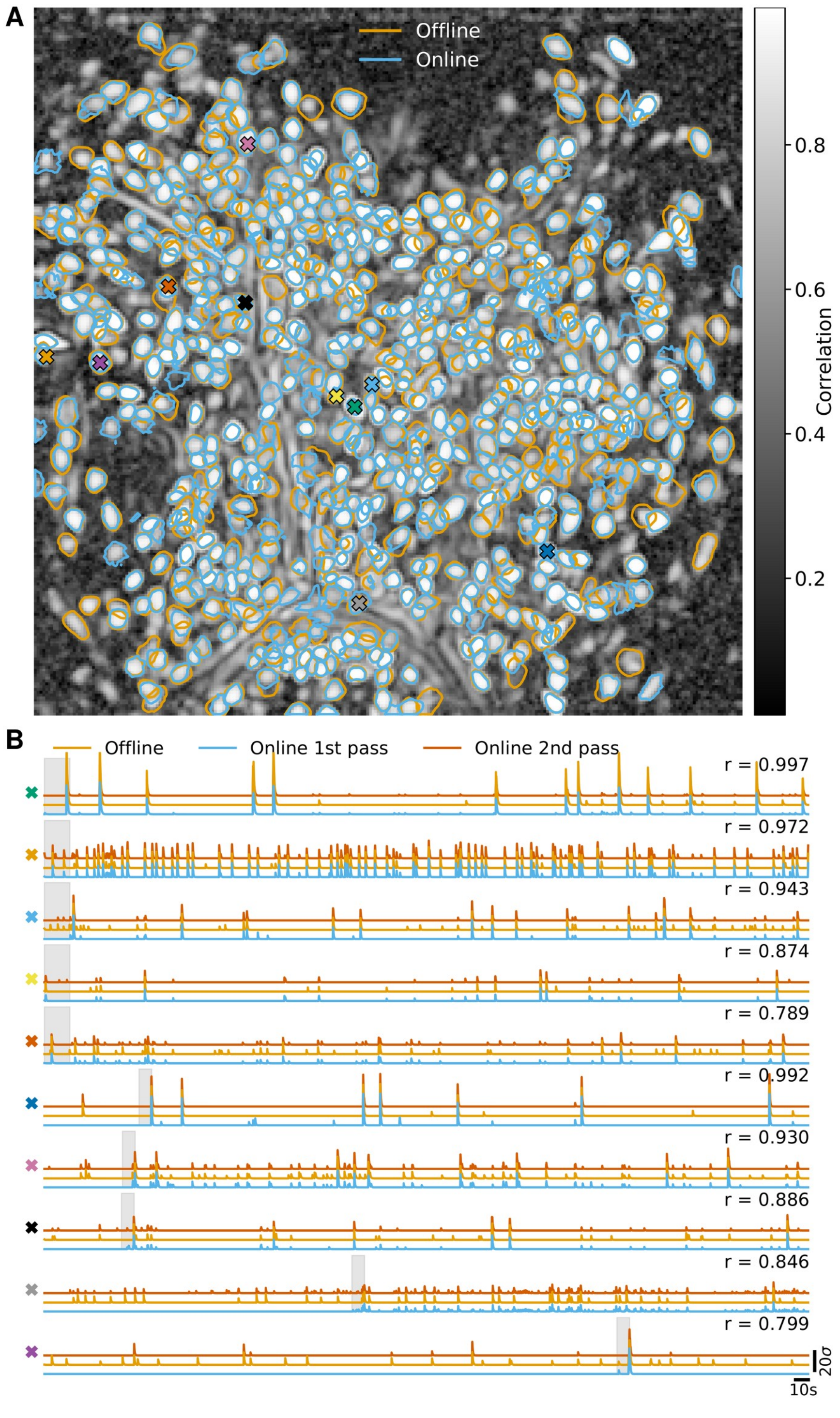

Fig 2. Comparison of OnACID-E with CNMF-E on data from neurons expressing GCaMP6f recorded *in vivo* in mouse dorsal striatum area. (A) Contour plots of all neurons detected by CNMF-E using patches (orange) and OnACID-E (blue), overlaid over the local cross-correlation image. Colors match the example traces shown in **(B)**, which illustrate the temporal components of ten example neurons detected by both implementations. The first five have been detected in the initialization phase, the last five during online processing. The gray shaded area shows the mini-batch OnACID-E used for the cell's initialization, thus the area's right border indicates at what frame the cell was initialized. The numbers to the upper right of each trace shows the correlation *r* between 'offline' and 'online 2nd pass'.

https://doi.org/10.1371/journal.pcbi.1008565.g002

actual cells. Ten example temporal traces are plotted in Fig 2B. The first five are from neurons that have been detected in the initialization phase, the last five during online processing. Not every neuron was detected immediately once it became active (blue traces); low activity events can be too weak to trigger detection as new component, but are accurately captured once the existence of the neuron has already been established.

Hence, when the data is analyzed after the experiment, e.g. when OnACID-E is used instead of CNMF-E for the sake of available computing resources (see below), one can perform a second online pass over the dataset, initialized with the results of the first pass, to recover the entire activity traces (red). The median correlation between the temporal traces of neurons detected by both implementations was 0.852 ± 0.008 (median ± standard error of the median).

We repeated the analysis on *in vivo* microendosopic data from neurons expressing GCaMP6s in prefrontal cortex of a freely behaving mouse. This second dataset consisted of 9000 frames at 15 Hz resolution (for further details refer to [6]). Analogous results to Fig 2 are presented in Fig 3. The F1-score between components detected by OnACID-E and CNMF-E was 0.899. The median correlation between the temporal traces of neurons detected by both implementations was 0.847 ± 0.022.

As third analysis we considered data from neurons expressing GCaMP6f recorded *in vivo* in mouse ventral hippocampus, cf. Fig 4. This third dataset consisted of 9000 frames at 15 Hz resolution (for further details refer to [6]). Here the FOV contained few enough neurons to label them manually, although one should be aware that in general human labelling is not perfect with different labelers often differing in their assessment [17]. We detected 23 neurons manually, whereas CNMF-E and OnACID-E detected 21 of these without any additional false positives (Fig 4A), yielding a F1-score of 0.955 when comparing either with the manually labeled components. Testing the quality of the inferred traces is more challenging due to the unavailability of ground truth data. As approximation to 'ground truth' we ran CNMF-E initialized with the centers of the manual annotation. We show ten example temporal traces in Fig 4B for CNMF-E, manually seeded CNMF-E and OnACID-E. The cosine similarity $\frac{\boldsymbol{a}^\top \boldsymbol{a}^*}{\|\boldsymbol{a}\|\|\boldsymbol{a}^*\|}$ between the neural shape obtained with manual initialization $\boldsymbol{a}^*$ and inferred neural shape $\boldsymbol{a}$ is reported in Fig 4C. The correlation between the corresponding temporal traces is shown in Fig 4D. Unsurprisingly, the traces obtained with the offline algorithm are more similar to the 'ground truth' traces than the traces obtained with the online algorithm, because the 'ground truth' traces were obtained by running the offline CNMF-E algorithm, but with manual instead of automatic initialization. The median correlation between the temporal traces of neurons detected by CNMF-E and 'ground truth' was 0.983 ± 0.005, for OnACID-E it was 0.938 ± 0.020.

We repeated the analysis on data from neurons expressing GCaMP6s recorded *in vivo* in mouse bed nucleus of the stria terminalis (BNST). This fourth dataset consisted of 4500 frames at 10 Hz resolution (for further details refer to [6]). We labeled 139 neurons manually, CNMF-E detected 126 of those and 16 additional components, OnACID-E detected 126 of the manually labeled ones and 15 additional components. Analogous results to Fig 4 are presented in Fig 5. The F1-score between components detected by CNMF-E and 'ground truth' was 0.897, for OnACID-E it was 0.904. The median correlation between the temporal traces of neurons detected by CNMF-E and 'ground truth' was 0.912 ± 0.017, for OnACID-E it was 0.854 ±0.026. A summary of the characteristics and results for each dataset is given in Table 2.

We also performed the comparison on the simulated data from [6], in order to compare not only the offline and online method with each other but both with underlying ground truth, see Fig 6. Both implementations detect all components (Fig 6A) with a perfect F1-score of 1. We again show ten example temporal traces in Fig 6B. The cosine similarity between true

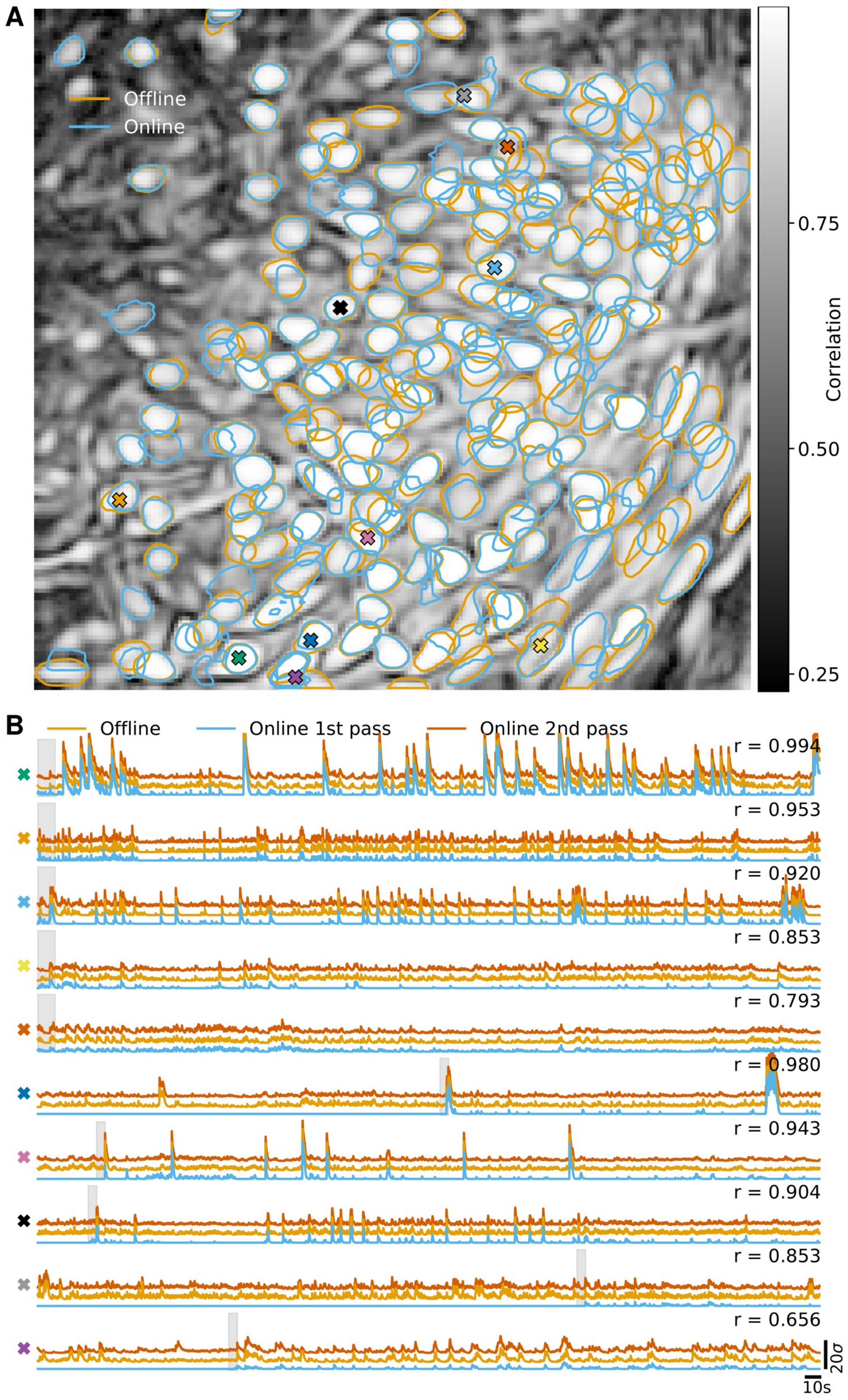

Fig 3. Comparison of OnACID-E with CNMF-E on data from neurons expressing GCaMP6s recorded *in vivo* in mouse prefrontal cortex. **(A)** Contour plots of all neurons detected by CNMF-E using patches (orange) and OnACID-E (blue), overlaid over the local cross-correlation image. Colors match the example traces shown in **(B)**, which illustrate the temporal components of ten example neurons detected by both implementations. The gray shaded area shows the mini-batch OnACID-E used for the cell's initialization, thus the area's right border indicates at what frame the cell was initialized. The numbers to the upper right of each trace shows the correlation *r* between 'offline' and 'online 2nd pass'.

https://doi.org/10.1371/journal.pcbi.1008565.g003

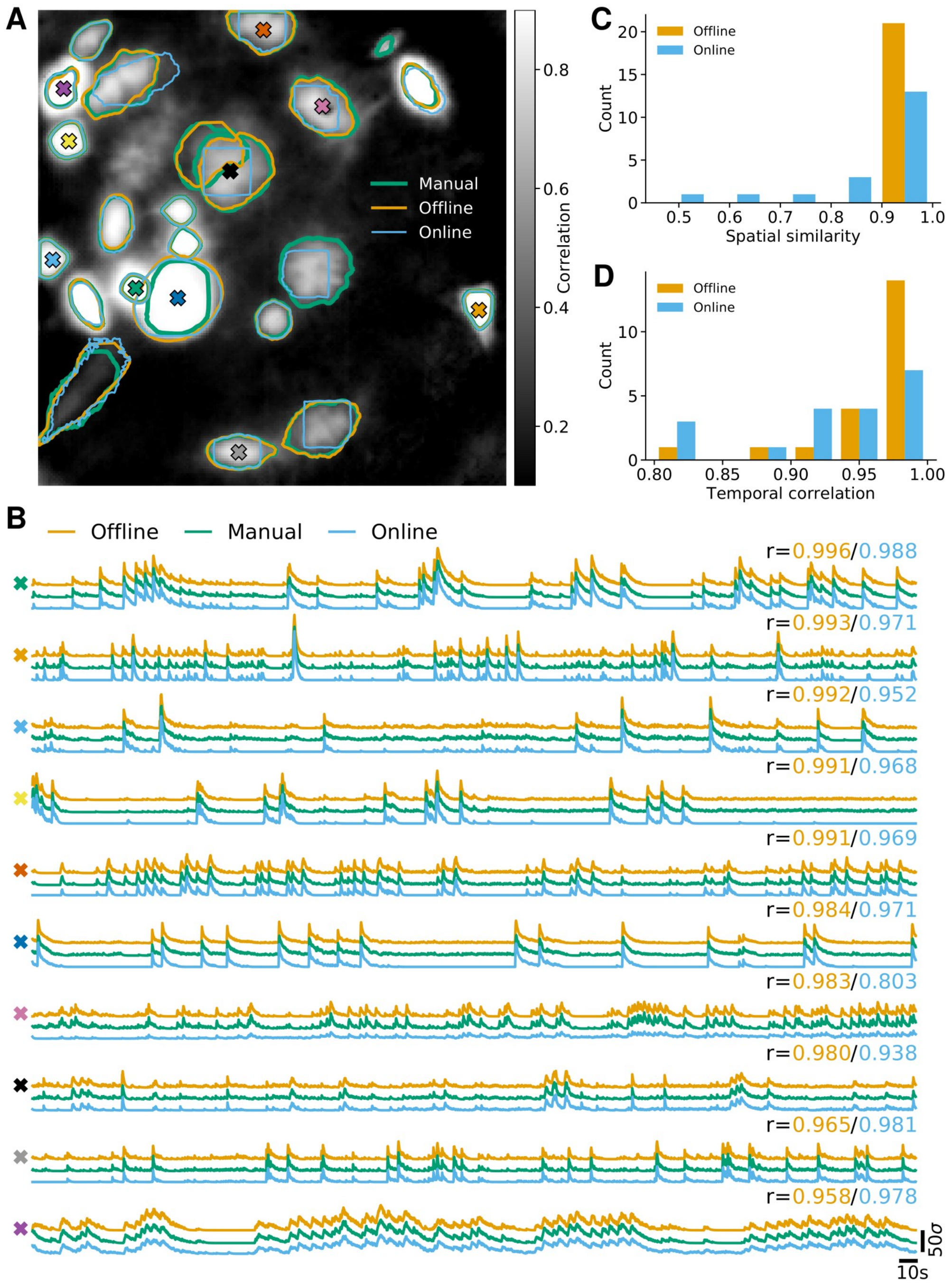

Fig 4. Comparison of OnACID-E with CNMF-E on data from neurons expressing GCaMP6f recorded *in vivo* in mouse ventral hippocampus. **(A)** Contour plots of ground truth neurons (green) as well as all neurons detected by CNMF-E (orange) and OnACID-E (blue), overlaid over the local cross-correlation image of the background-subtracted video data, with background estimated using CNMF-E with manual labeling. Colored symbols match the example traces shown in **(B)**, which illustrate the temporal components of ten example neurons detected by both implementations. The numbers to the upper right of each trace shows the correlation *r* between 'offline'/ 'online' and 'manual'. **(C)** Histogram of cosine similarities between inferred and true neural shapes. **(D)** Histogram of correlations between inferred and true neural fluorescence traces.

https://doi.org/10.1371/journal.pcbi.1008565.g004

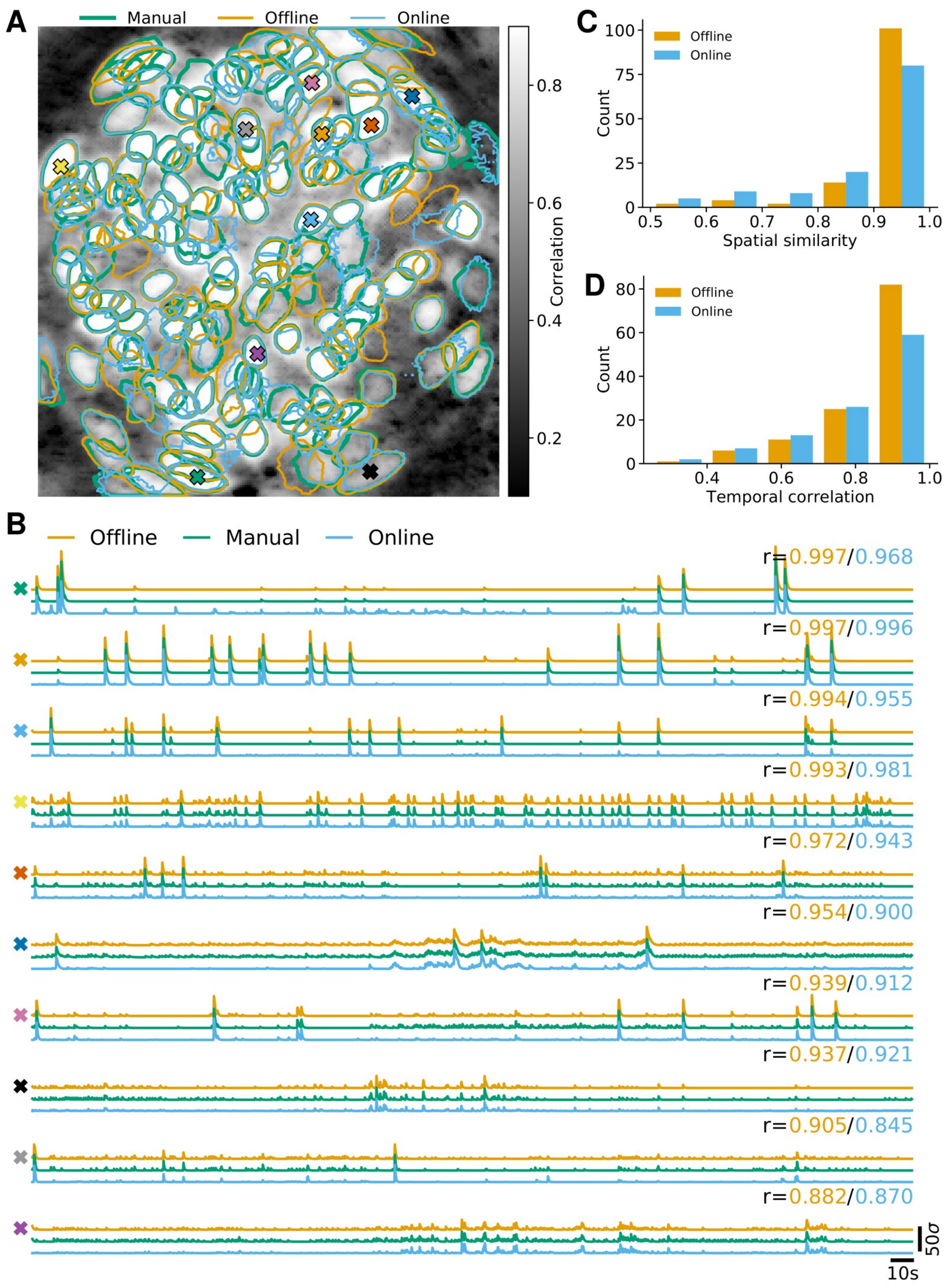

Fig 5. Comparison of OnACID-E with CNMF-E on data from neurons expressing GCaMP6s recorded *in vivo* in mouse bed nucleus of the stria terminalis (BNST). **(A)** Contour plots of ground truth neurons (green) as well as all neurons detected by CNMF-E (orange) and OnACID-E (blue), overlaid over the local cross-correlation image of the background-subtracted video data, with background estimated using CNMF-E with manual labeling. Colored symbols match the example traces shown in **(B)**, which illustrate the temporal components of ten example neurons detected by both implementations. The numbers to the upper right of each trace shows the correlation *r* between 'offline'/ 'online' and 'manual'. **(C)** Histogram of cosine similarities between inferred and true neural shapes. **(D)** Histogram of correlations between inferred and true neural fluorescence traces.

https://doi.org/10.1371/journal.pcbi.1008565.g005

Table 2. Dataset characteristics and performance measures.

Dataset		Striatum	PFC	Hippocampus	BNST
Size ($x \times y \times t$)		256 × 256 × 6000	175 × 184 × 9000	200 × 200 × 9000	199 × 203 × 4500
Rate [Hz]		10	15	15	10
F1-Score	CNMF-E	0.891	0.899	0.955	0.897
	OnACID-E			0.955	0.904
Correlation	CNMF-E	0.852 ± 0.008	0.847 ± 0.022	0.983 ± 0.005	0.912 ± 0.017
	OnACID-E			0.938 ± 0.020	0.854 ± 0.026
N	CNMF-E	582	171	21	142
	OnACID-E	554	185	21	141
% of frames processed in real-time	Tracking	100	100	100	100
	OnACID-E	84	94	90	92
	Ring-CNN	100	100	100	100

Individual entries for CNMF-E and OnACID-E denote comparison to 'ground truth' components (F1-Score) and traces (correlation) obtained by manual initialization of CNMF-E, shared entries denote direct comparison between CNMF-E and OnACID-E. While all methods process the datasets faster than real time *on average*, only for Tracking and Ring-CNN is the processing speed for *each* frame above the acquisition rate, whereas OnACID-E processes a high percentage of individual frames in real time.

https://doi.org/10.1371/journal.pcbi.1008565.t002

and inferred neural shape is reported in Fig 6C. While CNMF-E tends to capture the neural footprints more accurately, the inferred temporal components (that would be used in the subsequent analysis and are hence more important) are of similar quality, as the correlations with ground truth reveal (Fig 6D). The median correlation between the temporal traces of neurons detected by CNMF-E and ground truth was 0.9961 ± 0.0002, for OnACID-E it was 0.9932 ± 0.0005.

Computational performance of OnACID-E

We examined the performance of OnACID-E in terms of processing time and memory requirements for the analyzed dorsal striatum dataset (Fig 2) presented above. For the batch as well as the online algorithm we used the Python implementations provided by or added to CaImAn [17], respectively. OnACID-E has very limited memory requirements and can readily be run on a laptop. Thus, unless otherwise mentioned, the analysis was run on a laptop (MacBook Pro 13", 2017) with Intel Core i7-7567U CPU at 3.5 GHz (2 cores) and 16 GB of RAM running macOS Catalina.

The processing time of OnACID-E depends primarily on (i) the computational cost of tracking the temporal activity of discovered neurons, (ii) the cost of detecting and incorporating new neurons, and (iii) the cost of periodic updates of spatial footprints and background. Additionally, there is the one-time cost incurred for initialization. Fig 7A and S3(A) Fig show the cost of each of these steps for one epoch of processing. Initialization was performed by running CNMF-E on the first 200 frames, hence the sudden jump at 200 processed frames in Fig 7A. The cost of detecting and incorporating new components remains approximately constant across time and is dependent on the number of candidate components at each time step. In this example three candidate components were used per frame. As noted in [17], a higher number of candidate components can lead to higher recall in shorter datasets at a moderate additional computational cost (see S4 Fig).

The cost of tracking components can be kept low due to simultaneous vectorized updates, and increases only mildly over time as more components are found by the algorithm, cf.

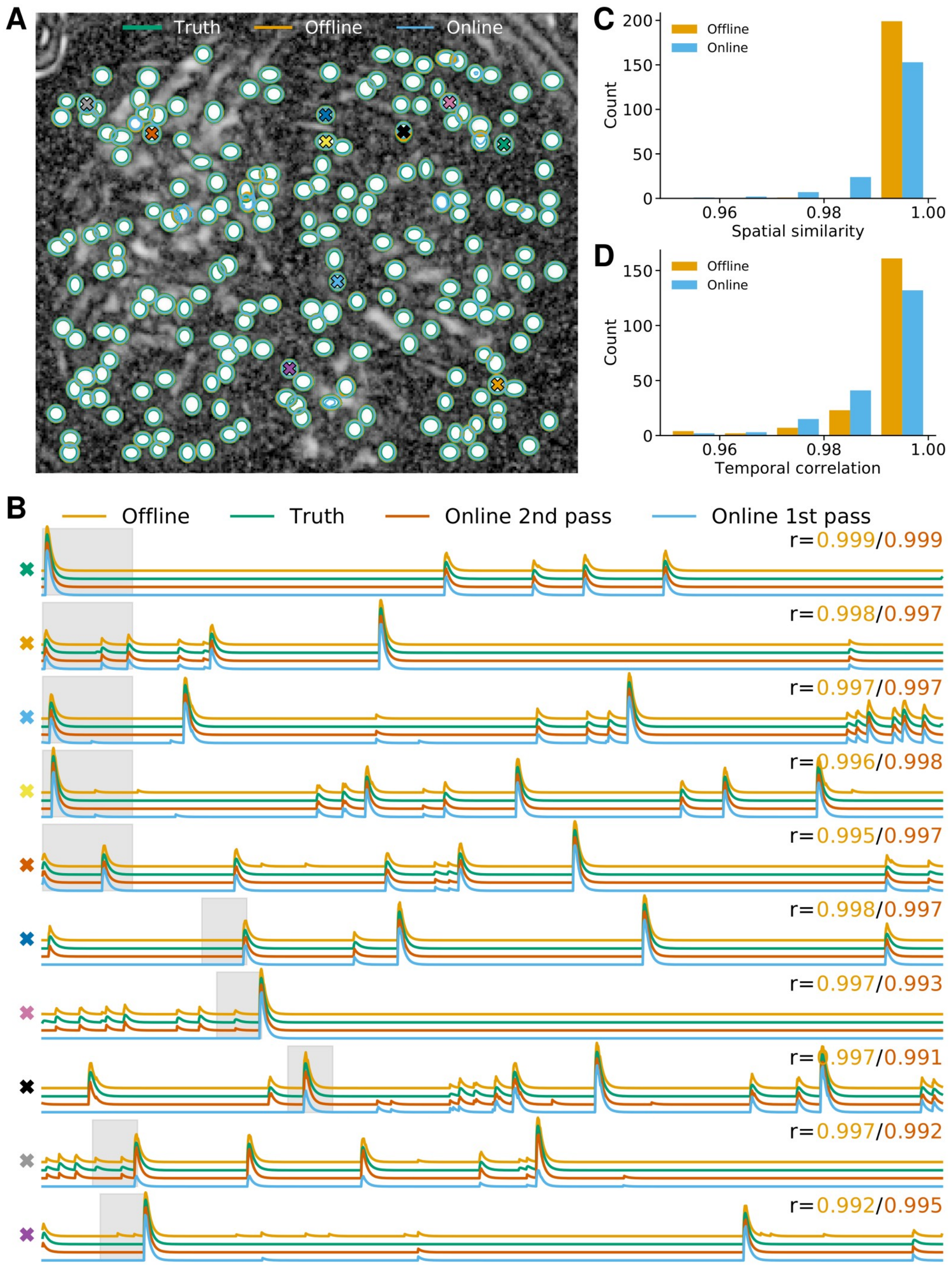

Fig 6. OnACID-E performs similar to CNMF-E in extracting individual neurons' activity from simulated data. **(A)** Contour plots of ground truth neurons (green) as well as all neurons detected by CNMF-E (orange) and OnACID-E (blue), overlaid over the local cross-correlation image. Colored symbols match the example traces shown in **(B)**, which illustrate the temporal components of ten example neurons detected by both implementations. The first five have been detected in the initialization phase, the last five during online processing. The gray shaded area shows the mini-batch OnACID-E used for the cell's initialization, thus the area's right border indicates at what frame the cell was initialized. The numbers to the upper right of each trace shows the correlation *r* between 'offline'/'online 2nd pass' and ground truth. **(C)** Histogram of cosine similarities between inferred and true neural shapes. **(D)** Histogram of correlations between inferred and true neural fluorescence traces.

https://doi.org/10.1371/journal.pcbi.1008565.g006

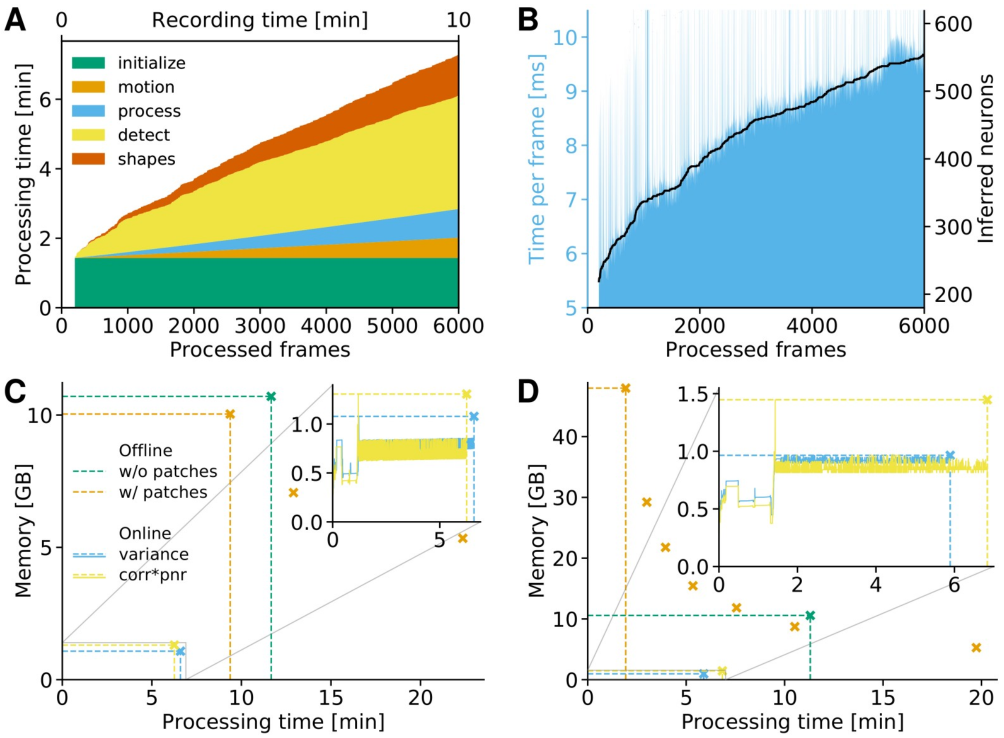

Fig 7. Computing resources of OnACID-E. The same dorsal striatum dataset from Fig 2 consisting of 6000 frames with a 256 × 256 FOV was used. **(A)** Cumulative processing time, separated by time for initialization (occurred only at the beginning), motion correction, tracking existing activity, detecting new neurons, and updating spatial footprints as well as background. **(B)** Cost of tracking neurons' activity scales linearly with the number of neurons. **(C)** Memory consumption of OnACID-E and CNMF-E. Markers and dashed lines indicate peak memory and overall processing time. Solid lines in the inset depict memory as function of time, and show that the required memory does not increase with the number of recorded frames. Offline processing using CNMF-E was performed with or without patches, online processing using OnACID-E with variance or corr⊙pnr summary image, cf. Methods. The orange markers show peak memory and overall processing time when the number of parallel processes is varied (4, 2 and 1), illustrating the time-memory trade off when processing in patches (more processes can lead to faster processing at the expense of additional memory requirements). **(D)** Analogous results as in (C) when using a single cluster node instead of the laptop, which enabled to process all patches simultaneously. Orange markers show peak memory and overall processing time for 16, 8, 6, 4, 3, 2 and 1 parallel processes.

https://doi.org/10.1371/journal.pcbi.1008565.g007

Fig 7B and S3(B) Fig. Finally, it is particularly noteworthy that the total processing time was smaller than the duration of the recording. If imaging was performed at a frame rate that is higher by some factor *x*, the cost of tracking would increase by this factor *x*, whereas the periodic updates of spatial footprints and background would still be performed every few seconds (20 s in our case). To keep the time spent on detecting new components invariant one can look for new components every *x*th frame. Thus the total required time is the sum of a high constant cost and the time for tracking that increases linearly with frame rate but is low.

Fig 7C shows the memory usage as function of processing time and compares to CNMF-E with or without splitting the FOV into patches. Sixteen patches of size 96x96 were used and processed in parallel. Due to the limited resources of the laptop (four threads, 16 GB RAM) not all, but merely up to four of the total sixteen patches, could be processed simultaneously in parallel. Fig 7C shows that whereas processing in patches was marginally faster and less memory consuming than processing the entire FOV, both are clearly outperformed with regard to computing time and memory requirements by OnACID-E. It required less memory than the size of the whole data, here 1.5 GB (for single-precision float), and about an order of magnitude less memory than CNMF-E. These results would be even more pronounced for longer datasets lasting not just minutes but hours because the memory consumption remains nearly constant as time progresses and is thus independent of the number of recorded frames.

Fig 7D repeats the analysis of Fig 7C, but using a single node of a linux-based (CentOS) cluster with Intel Xeon Platinum 8168 CPU at 2.7 GHz (24 cores) and 768 GB of RAM. This enabled to process all sixteen patches simultaneously in parallel. Processing can be faster using patches, however, this gain comes at the cost of high memory requirements compared to the raw data size and necessitates a powerful computing environment. These requirement can be mitigated at the expense of longer processing times by processing not all patches in parallel, as the additional orange markers in Fig 7D for 8, 6, 4, 3, 2 and 1 parallel processes show. Online processing on the cluster node took about the same time as on the laptop. OnACID-E strikes the best balance between memory consumption and processing time, making it in particular suitable for processing of long datasets without the need for high performance hardware.

Performance of the ring CNN approach

For comparison purposes we also tried the online analysis of the same dorsal striatum dataset, using the Ring-CNN background model (Eq 22) with two kernels of width 5 pixels. The model was trained on the first 500 frames (400 frames for training and 100 for validation) using a quantile loss function (Eq 26) with $q = 0.02$, using stochastic gradient descent with the ADAM optimizer. After initialization every frame was passed through the learned model to remove its background and was subsequently processed using the OnACID algorithm [17] with a rank-2 background, cf. Alg 2. During this phase, the background model was kept constant with no additional training, which resulted in faster processing. This was possible because the background model had already converged to a stable value during initialization because of the smaller number of parameters needed to be learned due to the large level of weight sharing. Moreover, the increased width of the filter increased the statistical power of the model making it less sensitive to outliers, and thus aiding faster convergence. Three epochs were used to process the dataset, with the third epoch being used only to track the activity of the existing neurons (and not to detect new components). After the online processing was done, the identified components were merged, and then screened for false positive using the tests employed in the CaImAn package [17].

Lacking a "ground truth" benchmark we compared its performance against the CNMF-E algorithm [6]. The results of the analysis are summarized in Fig 8. The algorithms displayed a high level of agreement (green contours in Fig 8A) with F1-score 0.848 (precision 0.81 and recall 0.888 treating the CNMF-E predictions as "ground truth"). While the agreement between the ring CNN appoach and CNMF-E was lower compared to the agreement between OnACID-E and CNMF-E, this cannot be readily interpreted as underperformance of the ring CNN approach. For example, the ring CNN approach identified several components that have a clear spatial footprint in the correlation image of the spatially filtered data (some examples are highlighted by the yellow arrows).

The computational performance of the ring CNN approach is shown in Fig 8B and 8C. In addition to a computing cluster node, an NVIDIA Tesla V100 SXM2 32GB GPU was deployed to estimate the background model and subsequently apply it. Overall initialization on the 500 frames required around 53s, roughly equally split between estimating and applying the background model, and performing "bare initialization" [17] on the background extracted to find 50 components and initialize the rank-2 background. After that processing was very fast for every frame (Fig 8B) with no computational bottlenecks (as opposed to OnACID-E where updating the background can take significant resources). Overall, the first epoch of processing

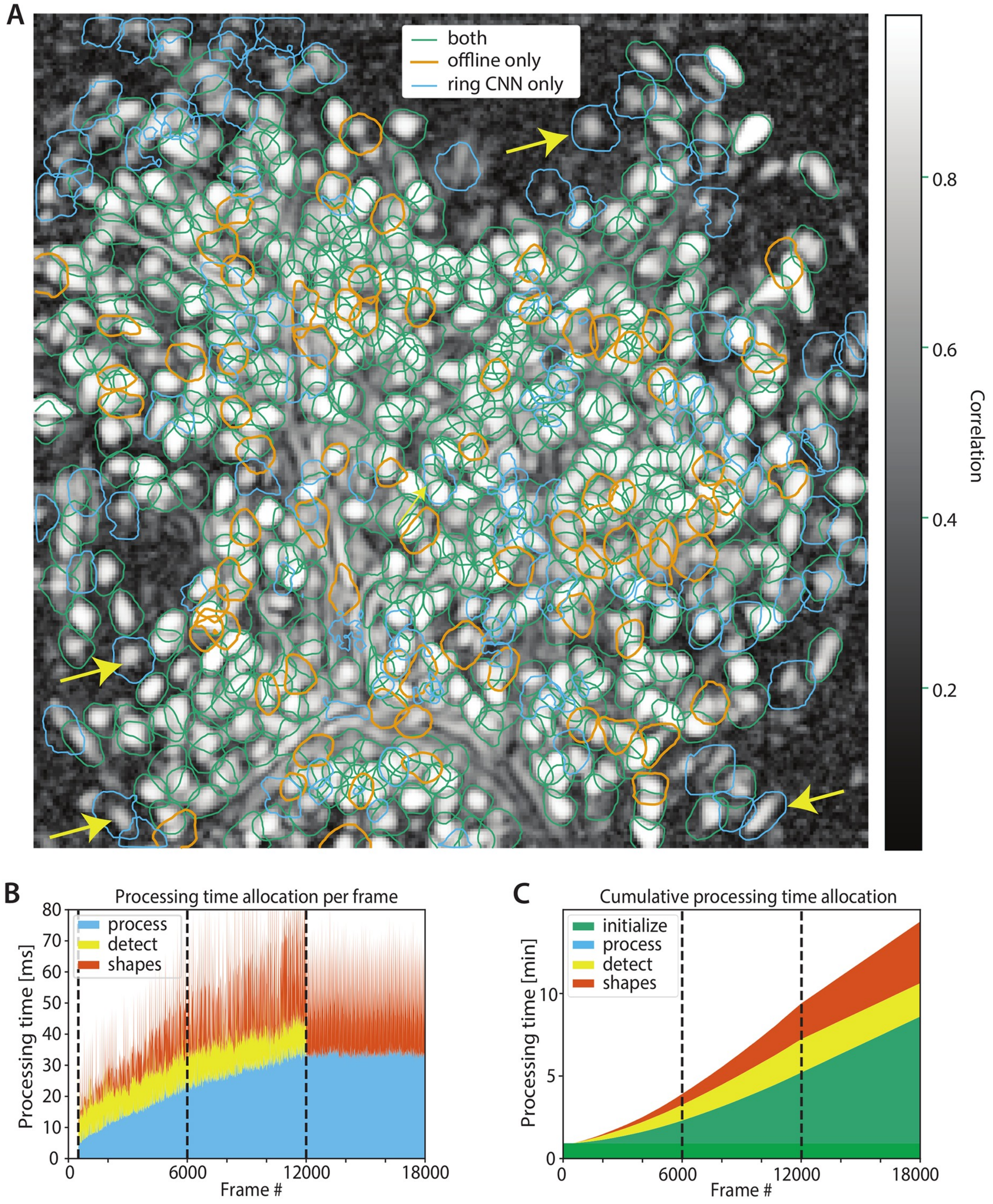

Fig 8. Performance of online approach using a ring CNN background model on the dorsal striatum dataset. **(A)** Contour plots of all neurons detected by the ring CNN approach and CNMF-E using patches overlaid over the local cross-correlation image. The two approaches have a high level of similarity (green contours, F1-score 0.845), with several components identified only by one algorithm (orange contours, CNMF-E only, blue contours, ring CNN only). At least some of the contours identified only by the ring CNN model appear to correspond to actual neurons (yellow arrows). Processing speed per frame **(B)** and cumulatively **(C)** for the ring CNN approach. Dashed lines indicate 1st, 2nd and 3rd epoch. By reducing the background extraction to a simple, GPU-implementable, filtering operation and estimating it only during initialization, the ring CNN approach can achieve high processing speeds for every frame **(B)**, and run a complete epoch on the data faster than OnACID-E **(C)**, cf. Fig 7A. Moreover, it can distribute the computational load evenly amongst all frames making it useful for real time applications.

https://doi.org/10.1371/journal.pcbi.1008565.g008

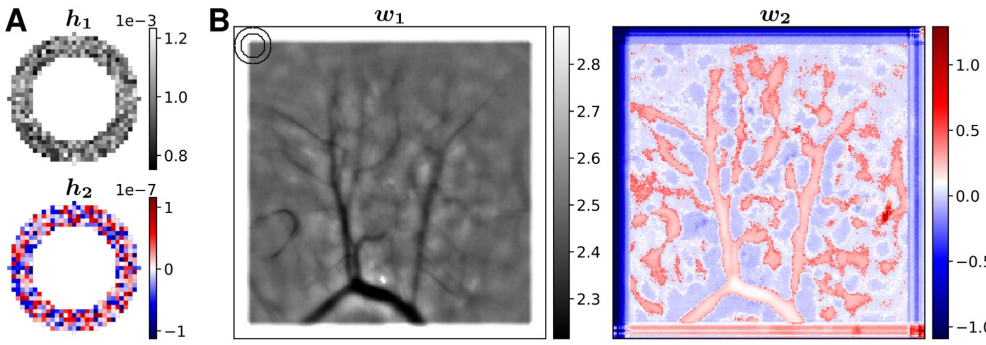

Fig 9. Parameters of the trained ring-CNN for the dorsal striatum dataset. **(A)** Convolution kernels of the first layer. **(B)** Pixel dependent weights of the second layer. The ring in the upper left corner indicates the size of the convolution kernels.

https://doi.org/10.1371/journal.pcbi.1008565.g009

was completed in 210s (Fig 8C), a factor of 2 improvement over OnACID-E (even without background updating for OnACID-E). A closer comparison between Figs 7A and 8C indicates that the ring CNN approach is faster than OnACID-E for detecting new components, but slower during "tracking". The reason for that, is that "tracking" in the ring CNN approach includes the background removing step (which requires data transfer to and from the GPU). However, once this step is done, no additional background treatment is required, which speeds up the detection step significantly. More importantly, this allowed a distributed update of shapes amongst all frames (Fig 8B) which kept the processing speed for *each* frame above the acquisition rate of 10Hz, thus achieving real time processing. Since the initialization step can be performed in mini-batches the GPU memory requirements remain limited. After that, online processing is deployed on a frame by frame basis which keep the memory requirements at similar levels compared to OnACID-E (S5 Fig).

Fig 9 shows the parameters of the trained ring CNN. As described in the Methods, the output of the network is invariant to rotations of the parameters. We used singular value decomposition, such that $\boldsymbol{h}_1$, $\boldsymbol{w}_1$ correspond to the left and right singular vectors of the larger singular value (14.1), and $\boldsymbol{h}_2$, $\boldsymbol{w}_2$ to the singular vectors of the smaller singular value (7.46×10^{-5}). The first convolutional kernel $\boldsymbol{h}_1$ shows the typical ring, and the weights $\boldsymbol{w}_1$ mostly capture pixel dependent differences in brightness, that are for example due to blood vessels. The second convolutional kernel $\boldsymbol{h}_2$ shows deviations from the typical ring, and the weights $\boldsymbol{w}_2$ where those deviations are applied. The much lower numerical values of $\boldsymbol{h}_2$ and $\boldsymbol{w}_2$ compared to h_1 and $\boldsymbol{w}_1$ reveal, that here the point spread function (PSF) was largely invariant over the entire FOV. In cases where the PSF varies when compared across the full FOV the number of rings K can be adjusted to the diversity of local environments.

The ring CNN approach introduced a new background model that was inspired by the ring model of CNMF-E/OnACID-E. It relies on removing the cumbersome background first, followed by an algorithm that has already been established for 2-photon data such as CNMF or OnACID. Similarly, Min1pipe [7] also relies on turning the imaging data into a stack of background-free frames as the first step, but uses morphological opening to estimate the background. In order to compare these methods, that use different background models, to underlying ground truth we considered again the simulated data from [6]. As a simple baseline we also included spatial high-pass filtering using a second order Butterworth filter with cut-off frequency of 6 inverse pixels. Fig 10A shows the local cross-correlation image of the data after removing the background for the four background models. While each method bring out all neurons that are not, or barely, visible in the raw video data (Fig 3 in [6]), the background

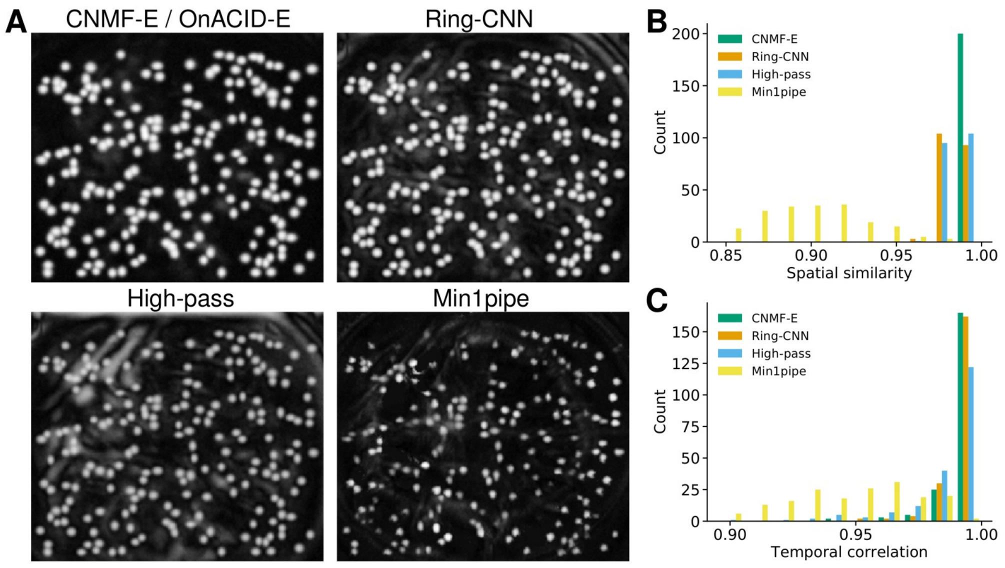

Fig 10. Comparison of background models on simulated data. **(A)** Local cross-correlation images of the background-subtracted video data. **(B)** Histogram of cosine similarities between inferred and true neural shapes. **(C)** Histogram of correlations between inferred and true neural fluorescence traces.

https://doi.org/10.1371/journal.pcbi.1008565.g010

model of CNMF-E visually captures the true background best. However, a good background estimate is just a means to enable extraction of accurate temporal traces that would be used in the subsequent analysis and are hence most important. To investigate how the background model influences the latter, we ran CNMF, initialized with the true neural centers, on these back-ground subtracted data, to obtain the temporal traces and updated spatial footprints. The cosine similarities between true and inferred neural shapes are reported in Fig 10B, with high values for the ring model of CNMF-E/OnACID-E, similar ones for Ring-CNN and high-pass filtering, and lower ones for morphological opening (Min1pipe). The inferred temporal components are of similar quality for CNMF-E and Ring-CNN, as the correlations with ground truth reveal (Fig 10C). The traces obtained with spatial high-pass filtering and morphological opening rank a close third and distant fourth respectively. The median correlation between ground truth and the temporal traces of neurons detected was 0.9970 ± 0.0003 for CNMF-E, 0.9942 ± 0.0004 for Ring-CNN, 0.9914 ± 0.0005 for high-pass filtering, and 0.9460 ± 0.0036 for morphological opening. We repeated the analysis on real data using the dorsal striatum data, cf. S6 Fig. Due to the lack of ground truth we compared to the results obtained with CNMF-E. Again, the Ring-CNN outperforms high-pass filtering.

Real time processing

One compelling reason to use online processing, not just for data streams but also for already recorded data, is that it circumvents the computational demands of offline processing. Even more impactful is its application to real time processing in closed loop experiments [4], for example combining imaging with optogenetic manipulation [30], which recently became technically possible also for 1-photon microendoscopes [31, 32]. Here we describe three approaches, cf. Fig 11A, for processing microendoscopic data in real time. We again considered the dorsal striatum data (Fig 2) and processed it using the MacBook Pro laptop.

The first, and arguably simplest, approach, denoted as “Tracking”, is to have a sufficiently long initialization phase to identify all ROIs. We processed an initial batch of 3000 frames

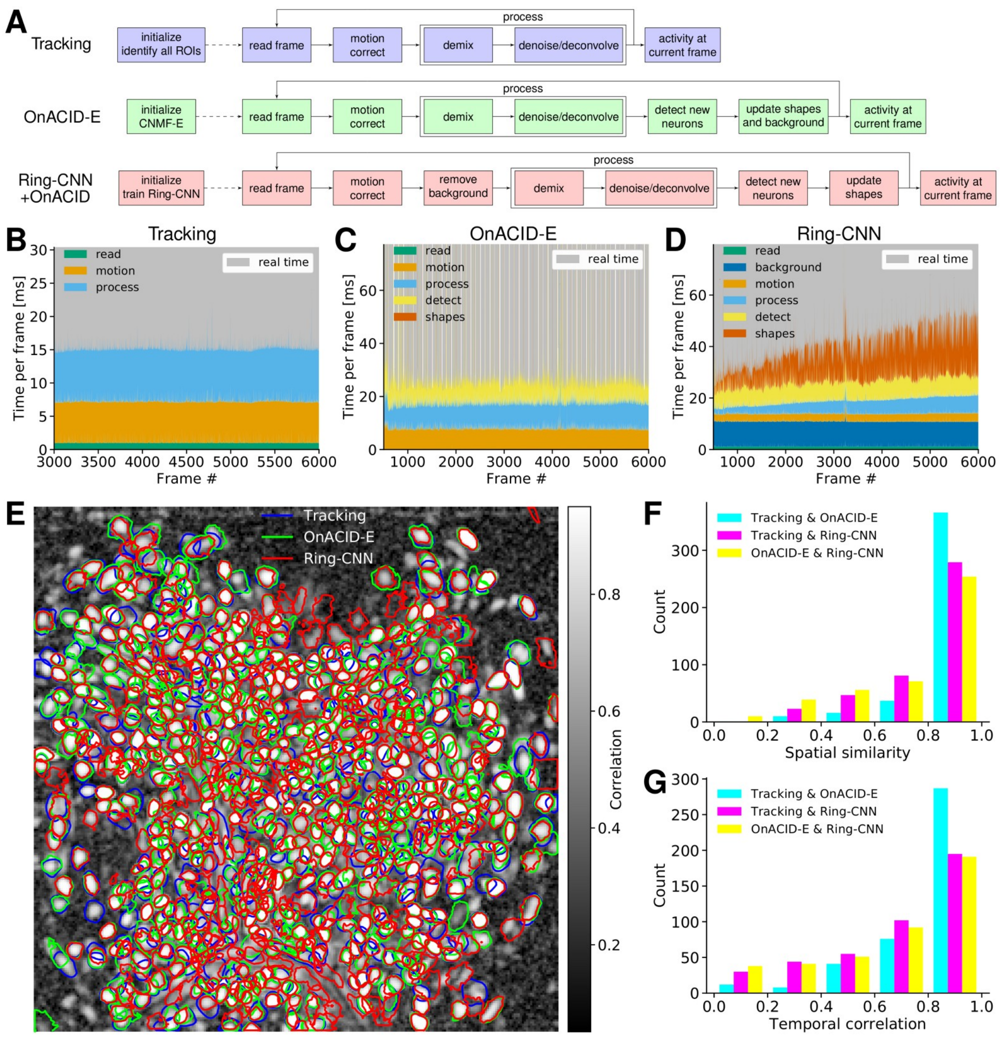

Fig 11. Online processing in real time. **(A)** Flow-charts for three online processing pipelines. **(B-D)** Time per frame for (**B**) tracking pre-identified ROIs, (**C**) OnACID-E, and (**D**) Ring-CNN, separated by time to read the frame, motion correct it, process it (i.e. demix overlapping components and deconvolve temporal trace), and, where applicable (OnACID-E and Ring-CNN), detect new neurons, update spatial footprints, and remove background. The upper limit of the vertical axis was set to twice the average total time per frame. The gray shading indicates the frames that are processed in real time. **(E)** Contour plots of all neurons detected by CNMF-E on a sufficiently long initial batch (Tracking), OnACID-E, and the Ring-CNN approach overlaid over the local cross-correlation image. **(F)** Histogram of pairwise cosine similarities of neural shapes for neurons detected using tracking, OnACID-E, and Ring-CNN. **(G)** Histogram of pairwise correlations of neural fluorescence traces.

https://doi.org/10.1371/journal.pcbi.1008565.g011

(5 min) using CNMF-E, though other offline pipelines could likewise be used, to initialize the online method OnACID-E. Once the initial batch has been processed, the real time experiment begins, during which each frame has to be read from the camera, corrected for motion artifacts, and processed to track each neuron's activity. The latter entails demixing the fluorescence contributions of (potentially overlapping) ROIs and background, as well as denoising/deconvolving each neuron's temporal trace. Fig 11B breaks down these times for each frame. In an actual real time experiment one would read directly from camera, here we report the time to read the frames from the laptop's solid-state drive. For further speed ups one could spatially decimate the identified ROIs and the raw data frames, and still accurately recover denoised fluorescence traces and deconvolved neural activity [26].

The second approach, denoted as “OnACID-E”, is to have merely a short initialization phase (we used 500 frames) followed by online processing using OnACID-E that not only tracks the activity of already detected ROIs as the first approach does, but includes automatic detection of new components and updates neural shapes as well as background. The latter updates are performed every few seconds (20 s in our case). They are however so computationally costly that real time performance is lost for this and some subsequent frames, as visible in the periodic pattern of the gray shading in Fig 11C. Thus, instead of the usual waiting for the camera to provide the next image, there are few frames for which the images are acquired faster than processed. Therefore we use a separate computing process to acquire and add the images to a FIFO (First-In-First-Out) queue at regular time intervals. The main process reads the next image from this queue, and waits for it if the queue is empty. Although this approach is not fully real time, 84% of the 5500 frames that have been processed online have actually been processed in real time.

The third approach, denoted as “Ring-CNN”, is similar to the second in using merely a short initialization phase (we used 500 frames), but instead of OnACID-E it uses Ring-CNN and OnACID for online processing. The background removing step involves evaluation of the CNN. This can be expressed as a sparse matrix multiplication, see Eq (23), which is quickly evaluated even on a laptop, cf. Fig 11D. Because the real time experiment starts only after the initial batch has been processed, the CNN can in principle be trained on a laptop as well. However, usage of a GPU is still advisable for this step in order to reduce the time where the camera is idle, i.e. the time between recording the last frame of the initial batch and starting the real-time experiment, which can take about an hour on a laptop, few minutes on a GPU, and even less than a minute on the high-performance GPU used in the previous subsection. Here we reused the previously trained CNN and converted it into a sparse matrix. The CNN possibly generalizes across imaging sessions, but we did not have access to the imaging data of multiple sessions of the same animal to test this. If retraining is necessary, a GPU is needed in praxis, but it does not need to be high end. This approach avoids OnACID-E’s complicated interaction between background and neural shapes, which allows a distributed update of the latter amongst all frames, thus achieving real time performance for each frame.

While all three approaches yield similar results, the first two model the background using the same ring model of CNMF-E, which as expected yields more similar results for them, compared to the third that employs the Ring-CNN. Fig 11E shows that all three approaches detect similar components. When comparing to Tracking, the F1-score was 0.892 for OnACID-E and 0.805 for Ring-CNN. This also holds for the spatial (Fig 11F) and temporal (Fig 11G) similarity measures when performing pairwise comparisons of the three approaches.

Discussion

We presented an online method to process 1-photon microendoscopic video data. Our modeling assumptions are the same as in the popular offline method CNMF-E; however, our online formulation yields a more efficient yet similarly accurate method for the extraction of in vivo calcium signals. A major bottleneck for processing microendoscopic data has been the amount of memory required by CNMF-E. Our online approach solves this issue since it reduces the memory footprint from scaling linearly with the duration of the recording to being constant. We also provided an additional variant that uses a convolutional based background model that aims to exploit the location invariant properties of the point spread function. This approach enables the estimation of a stable background model by using just an initial portion of the data. As a result, it can lead to faster processing and also be coupled to 2-photon processing algorithms by using this model to remove the background from each frame as a preprocessing step. While the background model is invariant to brightness changes, if the background structure was to change in a more intricate way owing to drug delivery, optogenetic stimulation, or other experimental manipulations, the employed 2-photon processing algorithm, e.g. OnACID, can itself include background components that are capable of capturing fluctuations, or training the CNN could continue based on the incoming data stream.

For detecting centroids of new sources OnACID-E examines a static image. Following [21], such an image can be obtained by computing the variance across time of the spatially smoothed residual buffer. As an additional option to obtain a static summary image we added the computation of peak-to-noise ratio and local cross correlation across time of the residual buffer, following the proposal of [6]. For efficiency, this computation is performed online using incremental updates. While both options work very well in practice, different approaches for detecting neurons in static images or in a short residual buffer could potentially be employed here, e.g. dictionary learning [33], combinatorial clustering [34] or deep neural networks [35, 36]. However, these approaches likely come with higher computational cost, and —having been developed for offline processing—would probably need to be modified for data streams, and in the case of neural networks be retrained.

Similarly to [21], our current implementation screens the candidate components for quality using some quantitative measures and thresholds. For 2-photon data [17] suggested to use a neural net classifier instead for better accuracy. Training a neural network requires labelled data, which is currently not publicly available for 1-photon microendoscopic video data. Once labelled ground truth data is available, a neural network could be trained on it and OnACID-E be readily augmented to use this classifier. Such ground truth data would also enable to thoroughly benchmark different source extraction algorithms and their implementations.

Apart from enabling rapid and memory efficient analysis of microendoscopic 1-photon data, our online pipeline also facilitates closed-loop behavioral experiments that analyze data on-the-fly to guide the next experimental steps or to control feedback. After recording a short initial batch of data for about one minute and processing it to initialize the online method, one can start the closed loop real time processing experiment. Although we did not have access to perform real time analysis hooked to an actual experiment, we emulated the environment to the best of our abilities. The current implementation of OnACID-E is already faster than real time on average. On a per-frame basis the processing speed exceeds the data rate for the majority of frames, and only when the periodic updates of sufficient statistics, shapes, and background are performed can the speed drop below the data rate. In principle, speed gains could be obtained by performing these periodic updates, and computations that occur only sporadically (incorporating a new neuron), in a parallel thread with shared memory. We defer that to future work. This speed drop below the data rate can be ameliorated by using a larger initialization batch for OnACID-E. Once enough initial data has been seen and processed, the computationally expensive search for components as well as the spatial footprint and background updates can be turned off, because all regions of interest have been detected and their shapes as well as the background converged to stable values. Further, as presented, this compromise can be avoided altogether by endowing the background with a convolutional structure that enables faster convergence in the background estimation. This subsequently enables updating of spatial footprints in a distributed sense, while maintaining faster than real time processing rates at *every* frame by keeping the ability to detect and incorporate new components. In summary, the Ring-CNN model lends itself better to actual real-time processing, whereas OnACID-E closely resembles the popular CNMF-E algorithm and lends itself to situations where the computing resources are not sufficient to run CNMF-E, or for real-time processing with sufficiently long initialization phase.

We provide a Python implementation of our algorithm online within CaImAn, an open-source library for calcium imaging data analysis (https://github.com/flatironinstitute/CaImAn) [17]. In order to facilitate the use of the presented algorithms in real time experimental scenarios, we have provided a set of example files and a flexible multi-threaded computational infrastructure which can be adapted to a variety of experimental settings. We have embedded online computations in a thread that is executed in parallel, while data can be incrementally added to a First-in-First-Out thread-safe queue. This software engineering design enables the use of OnACID-E with any type of acquisition systems, ranging from USB camera acquisition to dedicated high speed cameras with optimized hardware interfaces. The final user or camera company only needs to implement a small thread which pipes the frames into the queue.

Supporting information

S1 Appendix. Algorithmic description. Pseudocode for the various steps of the online processing pipeline.
(PDF)

S1 Fig. Additional components detected by CNMF-E in the dorsal striatum data from Fig 2.
(TIF)

S2 Fig. Additional components detected by OnACID-E in the dorsal striatum data from Fig 2.
(TIF)

S3 Fig. Processing time of OnACID-E using the variance summary image. Analogous plots to Fig 7A and 7B, but using the energy for each pixel of the residual buffer to create the summary image instead of the Corr*PNR summary image (see Methods).
(TIF)

S4 Fig. Effect of initial batch size and number of considered candidate components. **(A)** Detected components for different sizes of the initial batch without adjusting other parameters. Components detected in the initial batch are shown as solid contours, those detected during online processing as dashed contours. **(B)** Number of detected components for the initial batch sizes considered in (A). **(C)** Number of detected components for a varying number of candidate components.
(TIF)

S5 Fig. Memory usage during real-time processing. Memory and number of neurons as function of processed frames for **(A)** Tracking, **(B)** OnACID-E, and **(C)** Ring-CNN + OnACID.
(TIF)

S6 Fig. Comparison of background models on the dorsal striatum data from Fig 2. **(A)** Local cross-correlation images of the background-subtracted video data. **(B)** Histogram of cosine similarities between inferred neural shapes and the ones obtained with CNMF-E. **(C)** Histogram of correlations between inferred neural fluorescence traces and the ones obtained with CNMF-E.
(TIF)

S1 Video. Depiction of OnACID-E. Top left: Raw data and cell contours of all until then identified components. Top right: Inferred activity (without background). Bottom left: Corr*PNR

summary image (see Methods) and accepted regions for new components (magenta squares). Bottom right: Reconstructed activity.
(MP4)

Acknowledgments

We would like to thank Kris Pan for useful discussions and Pengcheng Zhou for helpful comments on the manuscript.

Author Contributions

Conceptualization: Johannes Friedrich, Andrea Giovannucci, Eftychios A. Pnevmatikakis.

Formal analysis: Johannes Friedrich, Eftychios A. Pnevmatikakis.

Investigation: Johannes Friedrich, Eftychios A. Pnevmatikakis.

Methodology: Johannes Friedrich, Eftychios A. Pnevmatikakis.

Software: Johannes Friedrich, Andrea Giovannucci, Eftychios A. Pnevmatikakis.

Visualization: Johannes Friedrich, Eftychios A. Pnevmatikakis.

Writing – original draft: Johannes Friedrich, Andrea Giovannucci, Eftychios A. Pnevmatikakis.

Writing – review & editing: Johannes Friedrich.

References

1. Ghosh KK, Burns LD, Cocker ED, Nimmerjahn A, Ziv Y, El Gamal A, et al. Miniaturized integration of a fluorescence microscope. Nature Methods. 2011; 8(10):871. https://doi.org/10.1038/nmeth.1694 PMID: 21909102
2. Cai DJ, Aharoni D, Shuman T, Shobe J, Biane J, Song W, et al. A shared neural ensemble links distinct contextual memories encoded close in time. Nature. 2016; 534(7605):115. https://doi.org/10.1038/nature17955 PMID: 27251287
3. Resendez SL, Jennings JH, Ung RL, Namboodiri VMK, Zhou ZC, Otis JM, et al. Visualization of cortical, subcortical and deep brain neural circuit dynamics during naturalistic mammalian behavior with head-mounted microscopes and chronically implanted lenses. Nature Protocols. 2016; 11(3):566. https://doi.org/10.1038/nprot.2016.021 PMID: 26914316
4. Aharoni DB, Hoogland T. Circuit investigations with open-source miniaturized microscopes: past, present and future. Frontiers in Cellular Neuroscience. 2019; 13:141. https://doi.org/10.3389/fncel.2019.00141 PMID: 31024265
5. 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
6. Zhou P, Resendez SL, Rodriguez-Romaguera J, Jimenez JC, Neufeld SQ, Giovannucci A, et al. Efficient and accurate extraction of *in vivo* calcium signals from microendoscopic video data. eLife. 2018; 7:e28728. https://doi.org/10.7554/eLife.28728 PMID: 29469809
7. Lu J, Li C, Singh-Alvarado J, Zhou ZC, Fröhlich F, Mooney R, et al. MIN1PIPE: a miniscope 1-photon-based calcium imaging signal extraction pipeline. Cell Reports. 2018; 23(12):3673–3684. https://doi.org/10.1016/j.celrep.2018.05.062 PMID: 29925007
8. Pinto L, Dan Y. Cell-type-specific activity in prefrontal cortex during goal-directed behavior. Neuron. 2015; 87(2):437–450. https://doi.org/10.1016/j.neuron.2015.06.021 PMID: 26143660
9. Ziv Y, Burns LD, Cocker ED, Hamel EO, Ghosh KK, Kitch LJ, et al. Long-term dynamics of CA1 hippocampal place codes. Nature Neuroscience. 2013; 16(3):264. https://doi.org/10.1038/nn.3329 PMID: 23396101
10. Klaus A, Martins GJ, Paixao VB, Zhou P, Paninski L, Costa RM. The spatiotemporal organization of the striatum encodes action space. Neuron. 2017; 95(5):1171–1180. https://doi.org/10.1016/j.neuron.2017.08.015 PMID: 28858619

11. Markowitz JE, Gillis WF, Beron CC, Neufeld SQ, Robertson K, Bhagat ND, et al. The striatum organizes 3D behavior via moment-to-moment action selection. Cell. 2018; 174(1):44–58. https://doi.org/10.1016/j.cell.2018.04.019 PMID: 29779950

12. Yu K, Ahrens S, Zhang X, Schiff H, Ramakrishnan C, Fenno L, et al. The central amygdala controls learning in the lateral amygdala. Nature Neuroscience. 2017; 20(12):1680–1685. https://doi.org/10.1038/s41593-017-0009-9 PMID: 29184202

13. da Silva JA, Tecuapetla F, Paixão V, Costa RM. Dopamine neuron activity before action initiation gates and invigorates future movements. Nature. 2018; 554(7691):244. https://doi.org/10.1038/nature25457 PMID: 29420469

14. Cameron CM, Murugan M, Choi JY, Engel EA, Witten IB. Increased cocaine motivation is associated with degraded spatial and temporal representations in IL-NAc neurons. Neuron. 2019; 103:80–91. https://doi.org/10.1016/j.neuron.2019.04.015 PMID: 31101395

15. Besnard A, Gao Y, Kim MT, Twarkowski H, Reed AK, Langberg T, et al. Dorsolateral septum somatostatin interneurons gate mobility to calibrate context-specific behavioral fear responses. Nature Neuroscience. 2019; 22(3):436. https://doi.org/10.1038/s41593-018-0330-y PMID: 30718902

16. Campos CA, Bowen AJ, Roman CW, Palmiter RD. Encoding of danger by parabrachial CGRP neurons. Nature. 2018; 555(7698):617. https://doi.org/10.1038/nature25511 PMID: 29562230

17. Giovannucci A, Friedrich J, Gunn P, Kalfon J, Brown BL, Koay SA, et al. CaImAn an open source tool for scalable calcium imaging data analysis. eLife. 2019; 8:e38173. https://doi.org/10.7554/eLife.38173 PMID: 30652683

18. Packer AM, Russell LE, Dalgleish HW, Häusser M. Simultaneous all-optical manipulation and recording of neural circuit activity with cellular resolution in vivo. Nature Methods. 2015; 12(2):140–146. https://doi.org/10.1038/nmeth.3217 PMID: 25532138

19. 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

20. Zhang Z, Russell LE, Packer AM, Gauld OM, Häusser M. Closed-loop all-optical interrogation of neural circuits in vivo. Nature Methods. 2018; 15(12):1037. https://doi.org/10.1038/s41592-018-0183-z PMID: 30420686

21. Giovannucci A, Friedrich J, Kaufman M, Churchland A, Chklovskii D, Paninski L, et al. OnACID: online analysis of calcium imaging data in real time. In: Advances In Neural Information Processing Systems 30; 2017. p. 2381–2391.

22. Mairal J, Bach F, Ponce J, Sapiro G. Online learning for matrix factorization and sparse coding. Journal of Machine Learning Research. 2010; 11(2):19–60.

23. Friedrich J, Zhou P, Paninski L. Fast online deconvolution of calcium imaging data. PLoS Computational Biology. 2017; 13(3):e1005423. https://doi.org/10.1371/journal.pcbi.1005423 PMID: 28291787

24. 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. Journal of Neurophysiology. 2010; 104(6):3691–3704. https://doi.org/10.1152/jn.01073.2009 PMID: 20554834

25. Cichocki A, Phan AH. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. IEICE transactions on fundamentals of electronics, communications and computer sciences. 2009; 92(3):708–721. https://doi.org/10.1587/transfun.E92.A.708

26. Friedrich J, Yang W, Soudry D, Mu Y, Ahrens MB, Yuste R, et al. Multi-scale approaches for high-speed imaging and analysis of large neural populations. PLoS Computational Biology. 2017; 13(8):e1005685. https://doi.org/10.1371/journal.pcbi.1005685 PMID: 28771570

27. Koenker R, Bassett G. Regression quantiles. Econometrica. 1978; 46(1):33–50. https://doi.org/10.2307/1913643

28. Pnevmatikakis EA, Giovannucci A. NoRMCorre: An online algorithm for piecewise rigid motion correction of calcium imaging data. Journal of Neuroscience Methods. 2017; 291:83–94. https://doi.org/10.1016/j.jneumeth.2017.07.031 PMID: 28782629

29. Zhou P, Resendez SL, Rodriguez-Romaguera J, Jimenez JC, Neufeld SQ, Giovannucci A, et al. Data from: efficient and accurate extraction of in vivo calcium signals from microendoscopic video data. Dryad Digital Repository. 2018; https://doi.org/10.5061/dryad.kr17k PMID: 29469809

30. Vander Weele CM, Siciliano CA, Matthews GA, Namburi P, Izadmehr EM, Espinel IC, et al. Dopamine enhances signal-to-noise ratio in cortical-brainstem encoding of aversive stimuli. Nature. 2018; 563 (7731):397–401. https://doi.org/10.1038/s41586-018-0682-1 PMID: 30405240

31. Stamatakis AM, Schachter MJ, Gulati S, Zitelli KT, Malanowski S, Tajik A, et al. Simultaneous optogenetics and cellular resolution calcium imaging during active behavior using a miniaturized microscope. Frontiers in Neuroscience. 2018; 12:496. https://doi.org/10.3389/fnins.2018.00496 PMID: 30087590

32. de Groot A, van den Boom BJ, van Genderen RM, Coppens J, van Veldhuijzen J, Bos J, et al. NIN-scope, a versatile miniscope for multi-region circuit investigations. eLife. 2020; 9:e49987. https://doi.org/10.7554/eLife.49987 PMID: 31934857

33. Petersen A, Simon N, Witten D. Scalpel: Extracting neurons from calcium imaging data. The Annals of Applied Statistics. 2018; 12(4):2430. https://doi.org/10.1214/18-AOAS1159 PMID: 30510612

34. Spaen Q, Asín-Achá R, Chettih SN, Minderer M, Harvey C, Hochbaum DS. HNCcorr: A novel combinatorial approach for cell identification in calcium-imaging movies. eNeuro. 2019; 6(2). https://doi.org/10.1523/ENEURO.0304-18.2019 PMID: 31058211

35. Apthorpe N, Riordan A, Aguilar R, Homann J, Gu Y, Tank D, et al. Automatic neuron detection in calcium imaging data using convolutional networks. In: Advances in Neural Information Processing Systems; 2016. p. 3270–3278.

36. Soltanian-Zadeh S, Sahingur K, Blau S, Gong Y, Farsiu S. Fast and robust active neuron segmentation in two-photon calcium imaging using spatiotemporal deep learning. Proceedings of the National Academy of Sciences. 2019; 116(17):8554–8563. https://doi.org/10.1073/pnas.1812995116 PMID: 30975747