Supplementary material

Algorithm 1 FIOLA

Require: FIFO input/output queues q_i, q_o, template T, spatial footprint A, fluorescence signals V

$\textsc{GpuMotionCorrect}, \textsc{GpuNnls}, \textsc{InferSpikes}, \mathbf{c} \leftarrow \textsc{Initialize}(T, A, V)$ ▷ initialize algorithms

while q_i is not empty **do**

  $F \leftarrow \textsc{pop}(q_i)$ ▷ Thread safe Tensorflow queue

  $F \leftarrow \textsc{GpuMotionCorrect}(\mathrm{F})$

  $\mathbf{c} \leftarrow \textsc{GpuNnls}(F)$

  $\mathbf{s} \leftarrow \textsc{InferSpikes}(\mathbf{c})$

  $\textsc{push}(q_o, \mathbf{s})$ ▷ Thread safe Tensorflow queue

Algorithm 2 GpuMotionCorrect

Require: Frame and template $F, T \in \mathbf{R}^{m \times n}$ where m,n are the height and width of the frame, maximum allowed shifts ms_x, ms_y, matrix of ones $\mathbf{1}_{p,q} \in \mathbf{R}^{p \times q}$ where $p = m - 2ms_x, q = n - 2ms_y$.

if Initialization **then**

  $T_m^* \leftarrow T - \textsc{Mean}(T)$

  $T_v^* \leftarrow \textsc{Var}(T)$ ▷ Variance of the templace $T_v \in \mathbf{R}$

  $\mathcal{T}_m^* \leftarrow \mathrm{FFT}(T_m)$

$F_m \leftarrow F - \textsc{Mean}(F)$

$F_v \leftarrow \textsc{AvgPool}(F^2, \mathbf{1}_{p,q}) - \textsc{AvgPool}(F, \mathbf{1}_{p,q})^2$. ▷ Local variance of $F_v \in \mathbf{R}^{(2ms_x+1)\times(2ms_y+1)}$

$CC \leftarrow \textsc{Abs}(\mathrm{IFFT}(\mathrm{FFT}(F_m) \cdot \overline{\mathcal{T}_m}))$ ▷ Correlation: $\overline{x}$ is the complex conjugate of x

$CC \leftarrow CC([\frac{m}{2}] - ms_x : [\frac{m}{2}] + ms_x, [\frac{n}{2}] - ms_y : [\frac{n}{2}] + ms_y)$ ▷ [x] = floor(x)

$CC_n \leftarrow \frac{CC}{\sqrt{T_v F_v}}$ ▷ Normalized cross correlation coefficient

$s_x, s_y \leftarrow \textsc{EstimateFractionalShifts}(CC_n)$ ▷ Uses gaussian interpolant

$F \leftarrow \textsc{RigidTranslation}(F, s_x, s_y)$ ▷ Uses bilinear interpolation

return F

Algorithm 3 GpuNnls

Require: Frame $F \in \mathbf{R}^{m \times n}$ where m,n are the height and width of the frame, spatial footprints $A \in \mathbf{R}^{d \times K}$ where d is the total number of pixels which equals to $m \times n$, K is the number of components, initial activity estimate $\mathbf{c^{(0)}} \in \mathbf{R}^{K \times 1}$, n_{iter} iterations

$\mathbf{y_t} \in \mathbf{R}^{d \times 1} \leftarrow$ Flatten(F)
if Initialization **then**
    $\alpha^* = \left\| A^T A \right\|_2$
    $\Theta_1^* \leftarrow I_n - \frac{A^T A}{\alpha}$
    $\mathbf{m^{(0)}} \leftarrow \mathbf{c^{(0)}}$
$\theta_2 \leftarrow \frac{A^T \mathbf{y_t}}{\alpha}$
for $k = [1, ..., n_{iter}]$ **do**
    $\mathbf{c^{(k)}} \leftarrow [\Theta_1 \mathbf{m^{(k-1)}} + \theta_2]_+$ ▷ Projected Gradient
    $\mathbf{m^{(k)}} \leftarrow \frac{k-1}{k+2}(\mathbf{c^{(k)}} - \mathbf{c^{(k-1)}})$ ▷ Extrapolate Acceleration Parameter
    $k \leftarrow k + 1$
return $\mathbf{c^{(k)}}$

Algorithm 4 InferSpikes

Require: number of frames for initialization T_{init}, fluorescence signal $\mathbf{c}$,lag for median filter lag, half window size for spike template τ, number of frames for updating statistics T_p, rest of parameters

if Initialization **then**
    $\mathbf{t_0}, \mathbf{t}, \mathbf{t_{sub}}, \mathbf{t_s}, \mathbf{s}, \mathbf{z}, thresh, md \leftarrow$ Init($\mathbf{c}(1 : T_{init})$) ▷ Initialization
    $T \leftarrow T_{init}$
$T \leftarrow T + 1$
$\mathbf{t_0}(T) \leftarrow \mathbf{t_0}(T-1) + 0.995(\mathbf{c}(T) - \mathbf{c}(T-1))$ ▷ Detrend by DC blocker
$\mathbf{t}(T) \leftarrow \mathbf{t_0}(T) - md$
$T' \leftarrow T - lag$
$\mathbf{t_{sub}}(T') \leftarrow$ MedianFilt($\mathbf{t}, lag$) ▷ Extract subthreshold signal
$\mathbf{t}(T') \leftarrow \mathbf{t}(T') - \mathbf{t_{sub}}(T')$
$\mathbf{t_s}(T' - \tau) \leftarrow$ CrossCorr($\mathbf{t}(T' - 2w : T'), \mathbf{z}$) ▷ Template matching
if $\mathbf{t_s}(T' - \tau - 1) >$ Max($\mathbf{t_s}(T' - \tau - 2), \mathbf{t_s}(T' - \tau)$) **then** ▷ Find peak
    if $\mathbf{t_s}(T' - \tau - 1) > thresh$ **then**
        $\mathbf{s} \leftarrow$ Append($\mathbf{s}, T' - \tau - 1$)
if $mod(T - T_{init}, T_p) = 0$ **then**
    $thresh, md \leftarrow$ UpdateStatistics($\mathbf{t_0}, \mathbf{t_s}, \mathbf{s}$)
return $\mathbf{t_s}, \mathbf{s}$

Algorithm 5 Initialization

Require: fluorescence signal $\mathbf{c} \in \mathbb{R}^T$, lag for median filter lag, stringency parameter for *adaptive threshold* p_1 and p_2, half window size for spike template τ, rest of parameters

$\mathbf{t_0}(1) \leftarrow \mathbf{c}(1)$
for T $\leftarrow 2$ to Len($\mathbf{c}$) **do**
    $\mathbf{t_0}(T) \leftarrow \mathbf{t_0}(T-1) + 0.995(\mathbf{c}(T) - \mathbf{c}(T-1))$
$md \leftarrow$ Median($\mathbf{t_0}$)
$\mathbf{t} \leftarrow \mathbf{t_0} - md$
$\mathbf{t_{sub}} \leftarrow$ MedianFilt($\mathbf{t}, lag$)
$\mathbf{t} \leftarrow \mathbf{t} - \mathbf{t_{sub}}$
$\mathbf{s_1}, thresh \leftarrow$ AdaptiveThreshold($\mathbf{t}, p_1$)
$\mathbf{z} \leftarrow \frac{1}{n(\mathbf{s_1})} \sum_{i=1}^{n(\mathbf{s_1})} \mathbf{t}(\mathbf{s_1}(i) - \tau : \mathbf{s_1}(i) + \tau)$ ▷ Compute spike template
$\mathbf{t_s} \leftarrow$ CrossCorr($\mathbf{t}, \mathbf{z}$) ▷ Template matching
$\mathbf{s_2}, thresh \leftarrow$ AdaptiveThreshold($\mathbf{t_s}, p_2$)
return $\mathbf{t_0}, \mathbf{t}, \mathbf{t_{sub}}, \mathbf{t_s}, \mathbf{s_2}, \mathbf{z}, thresh, md$

Algorithm 6 AdaptiveThreshold

Require: Trace $\mathbf{t_s} \in \mathbb{R}^T$, stringency parameter p, and remaining parameters *params*

$\mathbf{p} \leftarrow \textsc{LocalMaxima}(\mathbf{t_s})$ ▷ Find peak heights of all local maxima
$\mathbf{x} \leftarrow \textsc{LinSpace}(\textsc{Min}(\mathbf{p}), \textsc{Max}(\mathbf{p}), \textit{params})$ ▷ Evenly spaced samples between min and max of peak heights
$\mathbf{P_{max}} \leftarrow \text{KDE}(\mathbf{p}, \mathbf{x})$ ▷ Estimated distribution of local maxima at points $\mathbf{x}$
$\mu \leftarrow \textsc{Median}(\mathbf{p})$
$j \leftarrow \textsc{Find}(\mathbf{x}(i) < \mu, \mathbf{x}(i+1) > \mu)$
$\mathbf{P_{noise}} \leftarrow \textsc{Zeros}(\textsc{Len}(\mathbf{P_{max}}))$ ▷ Create a zero vector same size as $\mathbf{P_{max}}$
$\mathbf{P_{noise}}(1:j) \leftarrow \mathbf{P_{max}}(1:j)$ ▷ Estimate noise distribution by symmetrization
if $2j \geq \textsc{Len}(\mathbf{P_{max}})$ **then**
    $\mathbf{P_{noise}}(j+1:\text{end}) \leftarrow \mathbf{P_{noise}}(j:2j - \textsc{Len}(\mathbf{P_{max}}) + 1)$
else
    $\mathbf{P_{noise}}(j+1:2j) \leftarrow \mathbf{P_{noise}}(j:1)$
$\mathbf{F_{max}} \leftarrow \textsc{CumSum}(\mathbf{P_{max}})$ ▷ Cumulative distribution function
$\mathbf{F_{noise}} \leftarrow \textsc{CumSum}(\mathbf{P_{noise}})$
$\mathbf{F_{max}} \leftarrow \mathbf{F_{max}}(\text{end}) - \mathbf{F_{max}}$
$\mathbf{F_{noise}} \leftarrow \mathbf{F_{noise}}(\text{end}) - \mathbf{F_{noise}}$
$\mathbf{g} \leftarrow (\mathbf{F_{max}})^p - (\mathbf{F_{noise}})^p$
$h \leftarrow \mathbf{x}(\textsc{ArgMax}(\mathbf{g}))$
$\mathbf{s} \leftarrow \textsc{LocalMaxima}(\mathbf{t_s}, h)$ ▷ All local maxima with height greater or equal to h
return $\mathbf{s}, h$

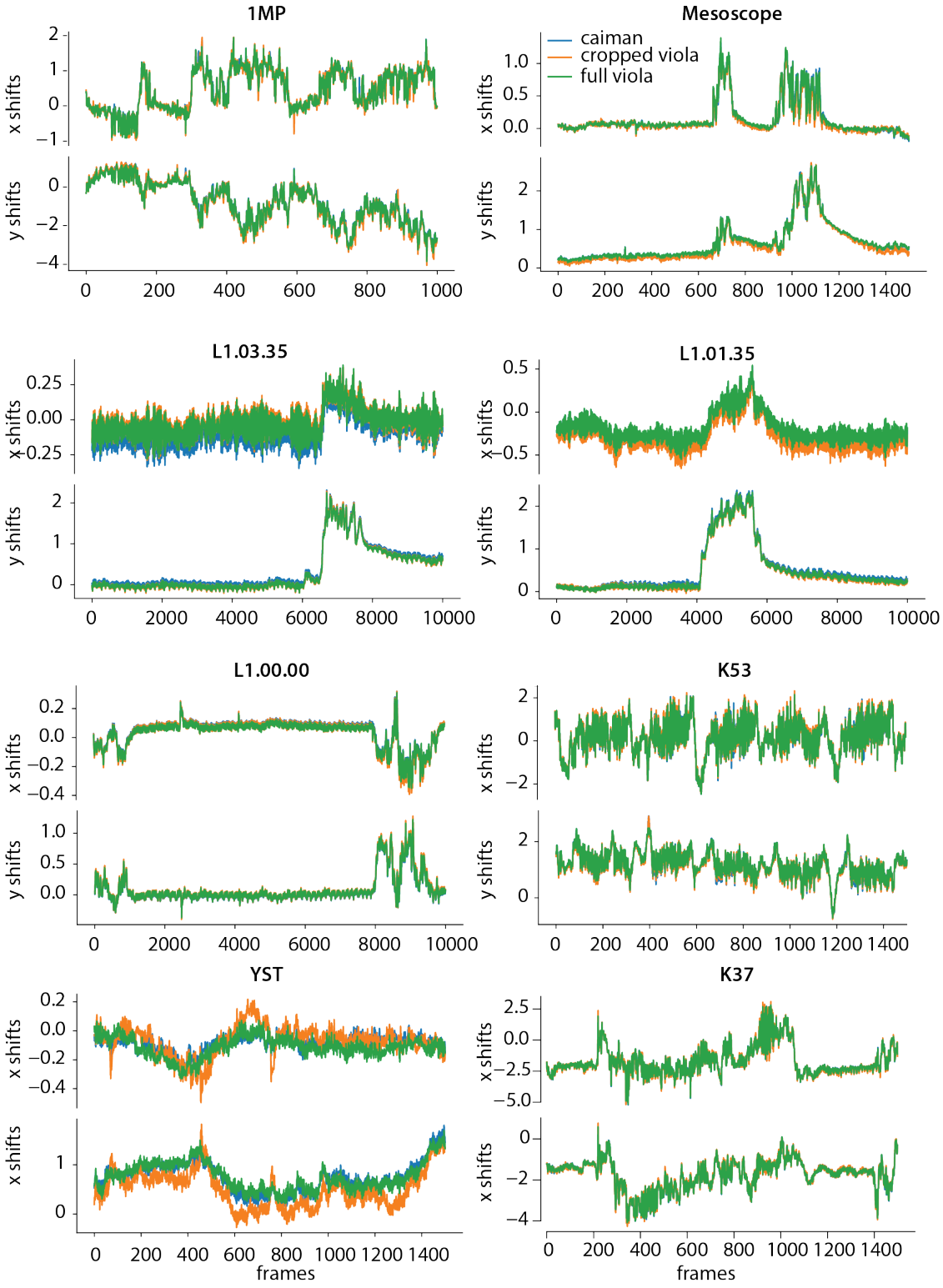

Figure 1: Comparison of shifts generated by FIOLA and by the NoRMCorre algorithm as implemented in CaImAn for all datasets. Shifts necessary to register the frame by a rigid translation across x and y predicted by CaImAn (blue line, ground truth) and FIOLA using 100% (green) and 50% (orange) of the frame.

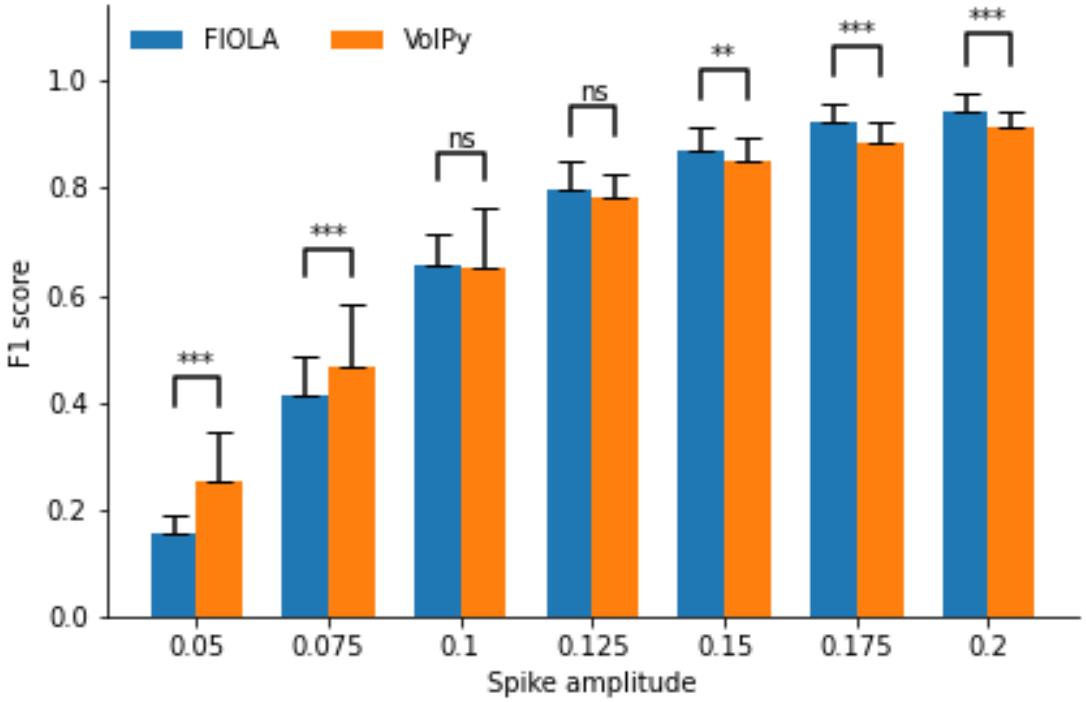

Figure 2: Comparison of FIOLA (blue) and VolPy (orange) on simulated data with non-overlapping neurons. Two-sided Wilcoxon signed-rank test were performed. *** $p \leq 0.001$,** $p \leq 0.01$, * $p \leq 0.05$, ns-not significant. Error bars are standard deviation.

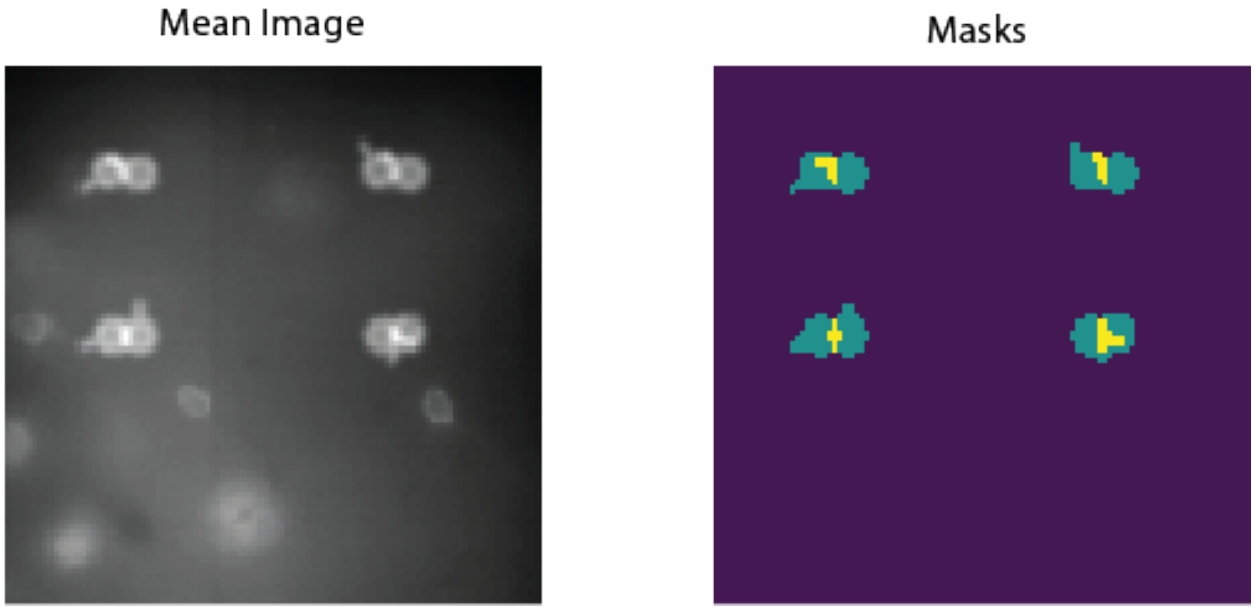

Figure 3: Simulated datasets including 8 overlapping neurons with 16% of overlap. Left: Mean image of the simulated datasets. Right: Corresponding masks of these neurons. Overlapping areas of two masks are shown in yellow.

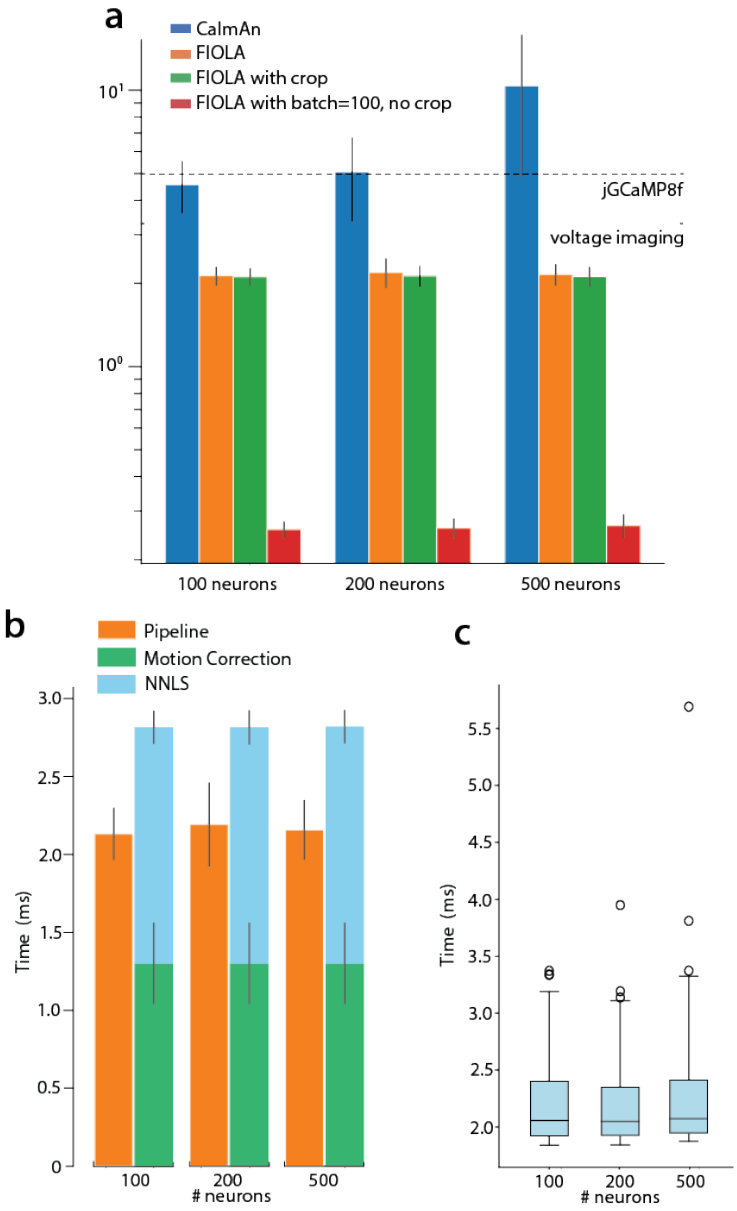

Figure 4: Evaluation of FIOLA computational speed for 256x256 movies. **a**. FIOLA's runtime (ms) per frame for 100, 200, and 500 neurons, compared to CaImAn (blue), FIOLA with a 128x128 crop, and FIOLA run in a batch of 100, without a crop. **b**. Comparison of the runtime of the full FIOLA pipeline (orange) and runtimes of the separate motion correction and NNLS models, added together (green and blue, respectively). **c**. Box-and-whisker plot showing FIOLA runtime per 256x256 frame, with a 128x128 central crop. The box represents times between the 5th and 95th quantiles, and the whiskers span from the 0.1st to the 99.9th quantile. Outliers are shown as circles.

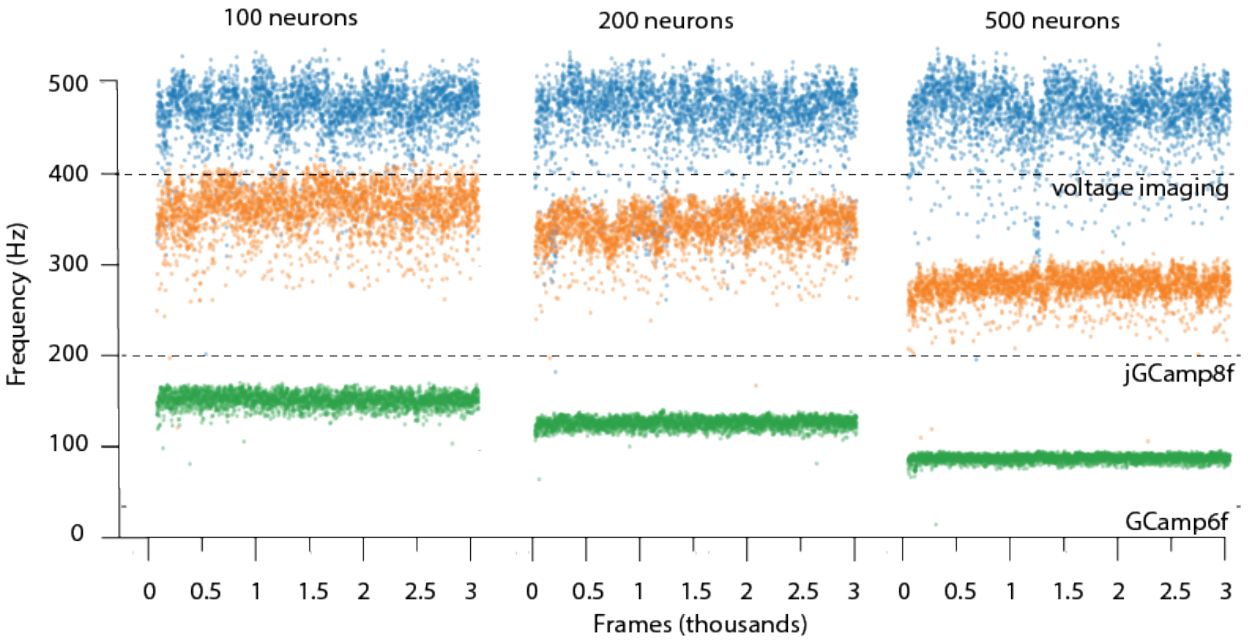

Figure 5: FIOLA online spike extraction speed performance. The rate for processing each frame is recorded with 500, 200, and 100 neurons, respectively, and three frame sizes: 256x256 (blue), 512x512 (orange), 1024x1024 (green)

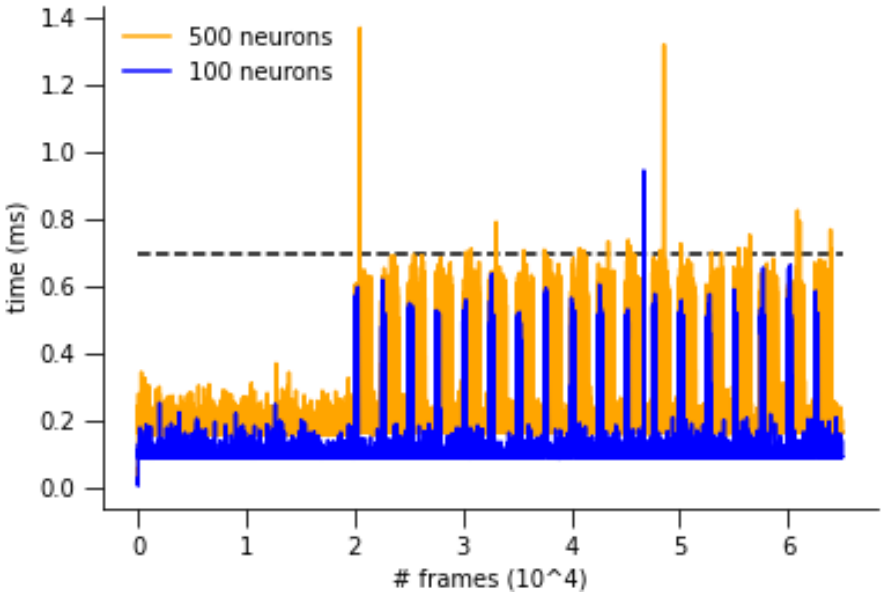

Figure 6: Speed for spike extraction algorithm. We ran online spike extraction algorithm in FIOLA on traces extracted from NNLS with 500 (yellow) and 100 (blue) neurons respectively. Processing 500 neurons took less than 700 μs (black dashed line) for most frames. Large 'spikes' in the figure are due to updating statistics step.