COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

# Computational Tutorial: Calcium Imaging Data Cell Extraction

Pengcheng Zhou

Department of Statistics
Columbia University

July 12th, 2017

# Outline

- Part I: Overview of methods for cell extraction
  - General methods for calcium imaging data
  - Constrained Nonnegative Matrix Factorization for microEndoscopic data (CNMF-E)
- Part II: Exercises

- Part I: Overview of methods for source extraction
  - **General methods for calcium imaging data**
  - Constrained Nonnegative Matrix Factorization for microEndoscopic data (CNMF-E)
- Part II: Exercises

# Calcium Imaging

spiking activity

fast, electrical

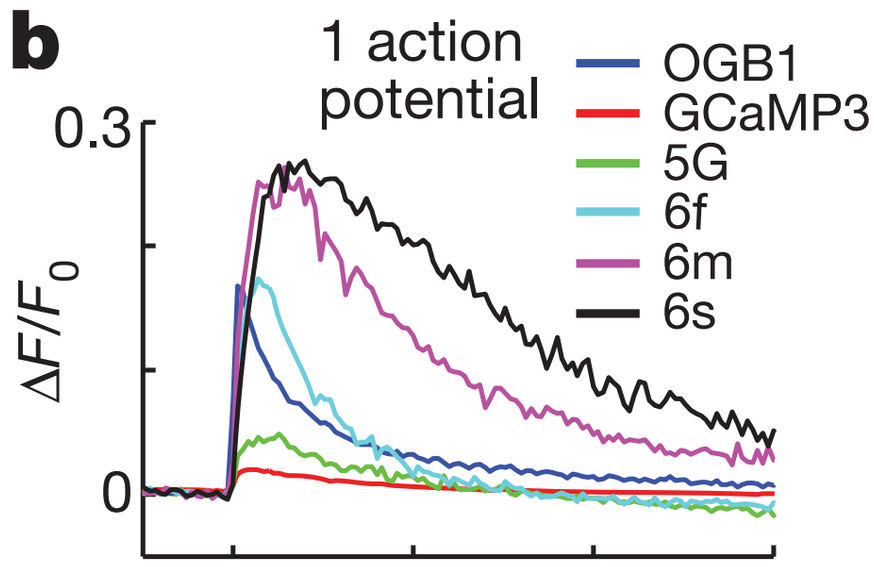
↓

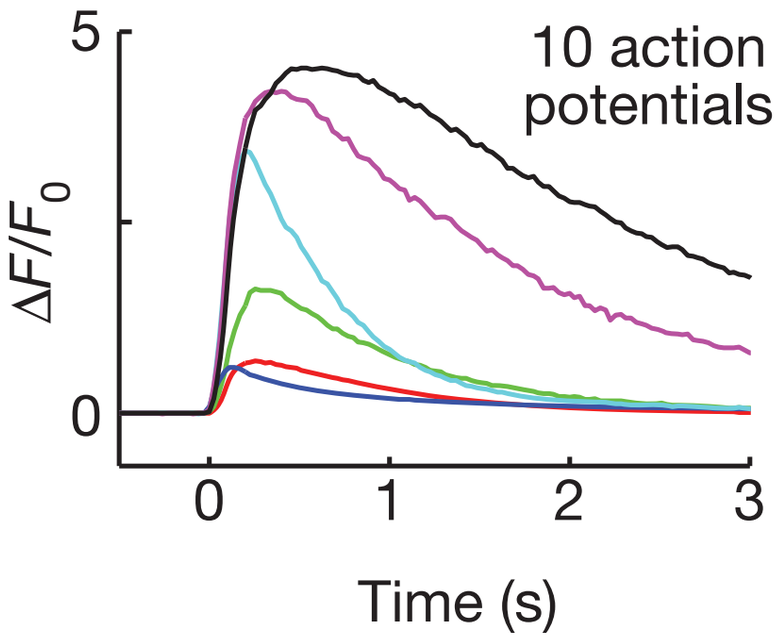
intracellular [$Ca^{2+}$]

↓

fluorescence intensity

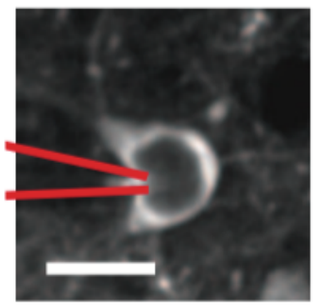
slow, optical




Chen et.al., Nature 2013

**a**



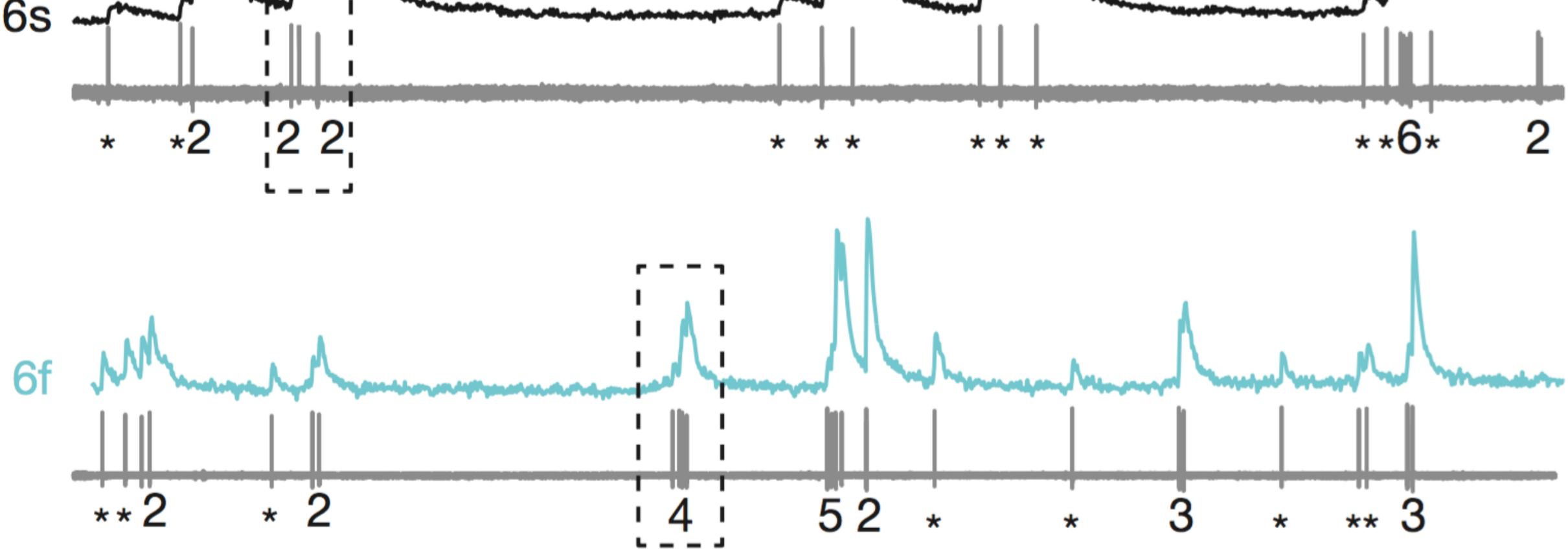
10 μm

* 1 action potential events

100%

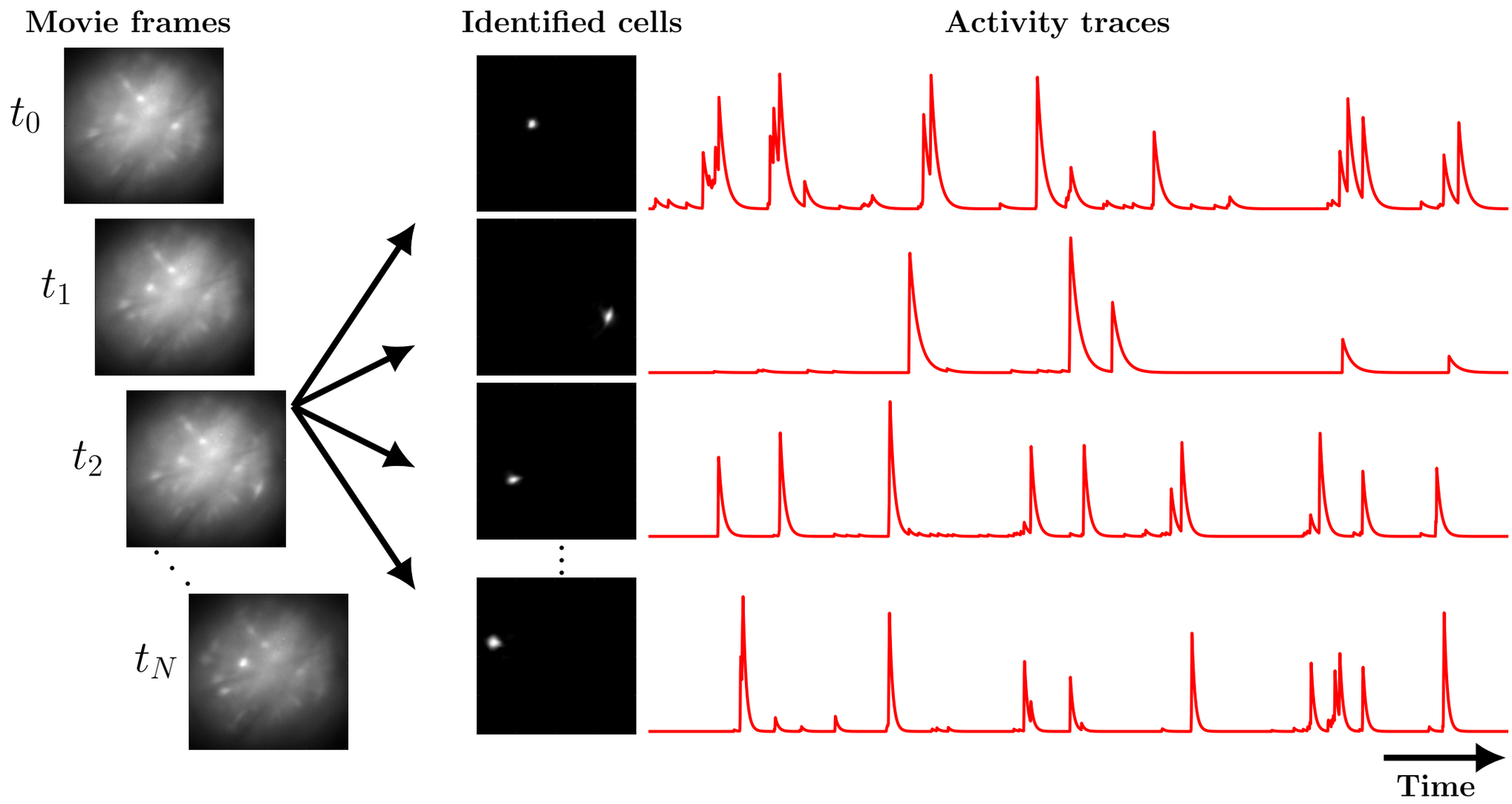
5 s

6s




Chen et.al., Nature 2013

# Cell Extraction Problem



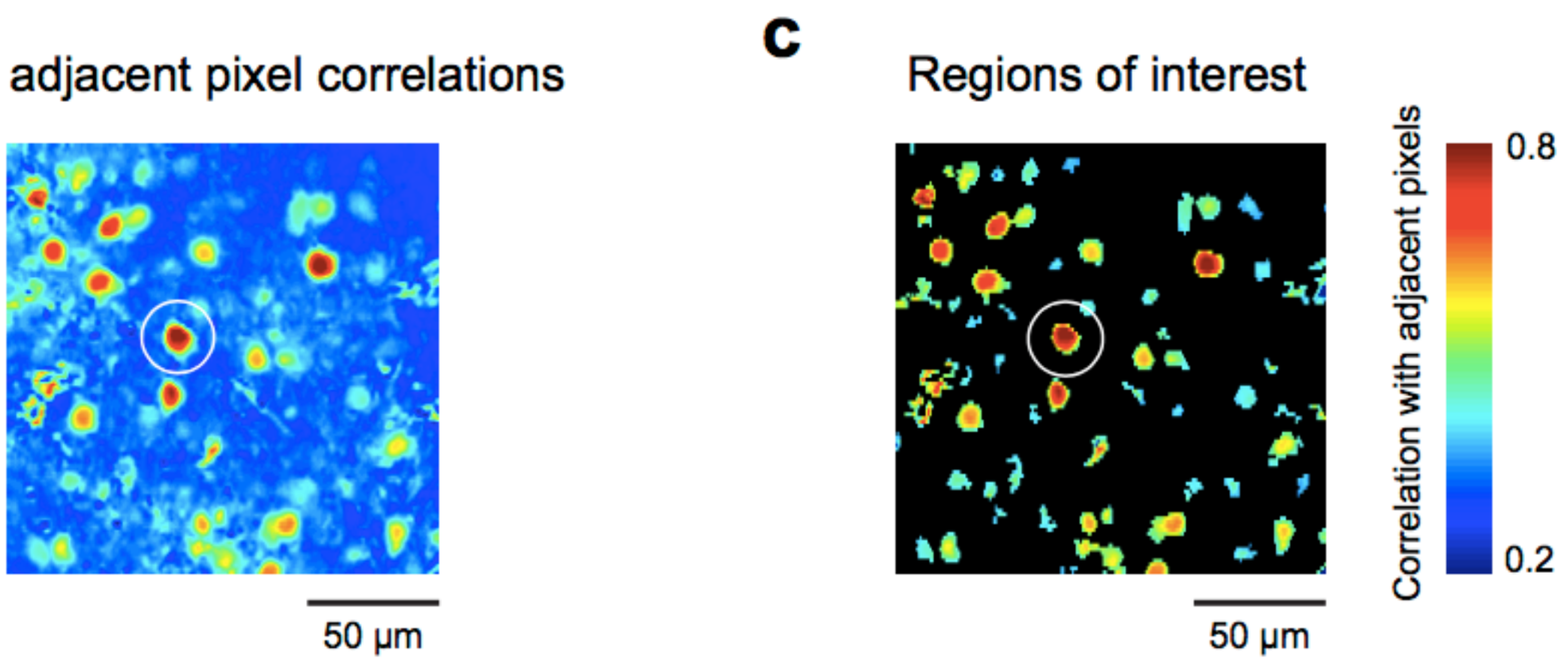

Given a video data, identify all neurons and extract their temporal traces.

# Two Approaches

- ROI analysis
- Matrix factorization approach

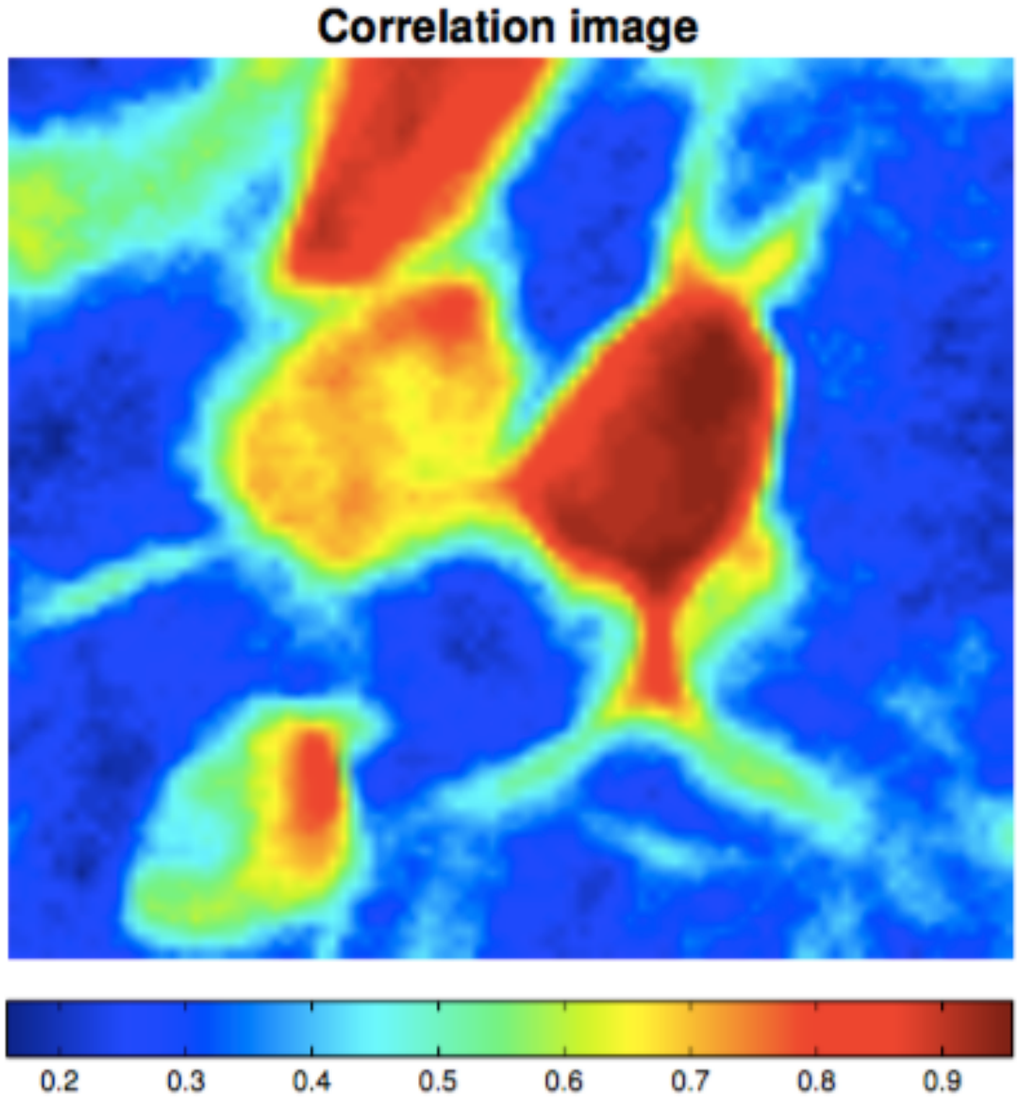
# ROI analysis

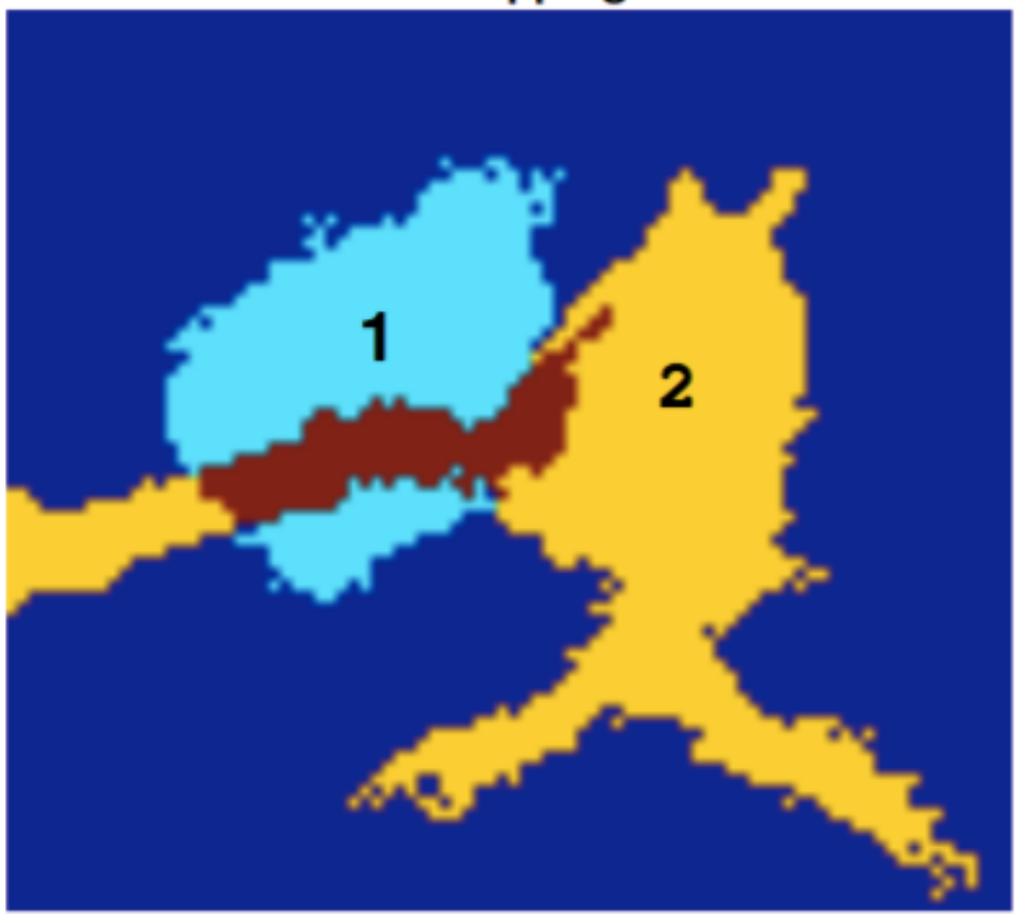
1. draw neuron's ROI manually or automatically
2. extract neural activity using the summed fluorescence intensities within each ROI




Smith & Hauser, Nature Neuroscience, 2010

# Problems

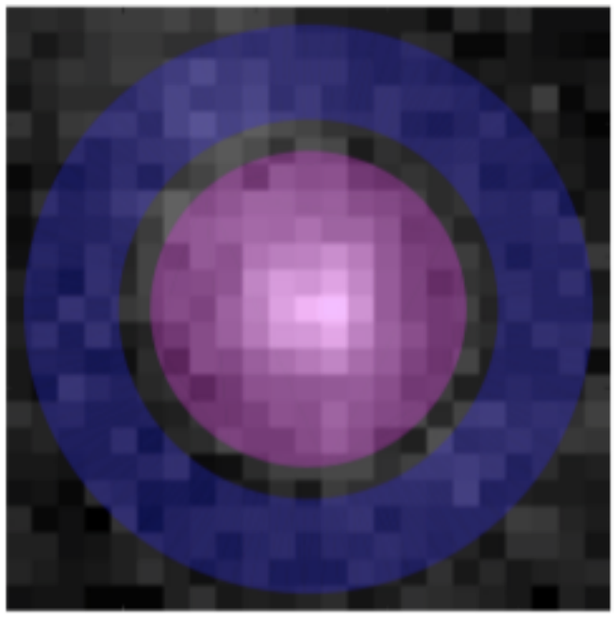


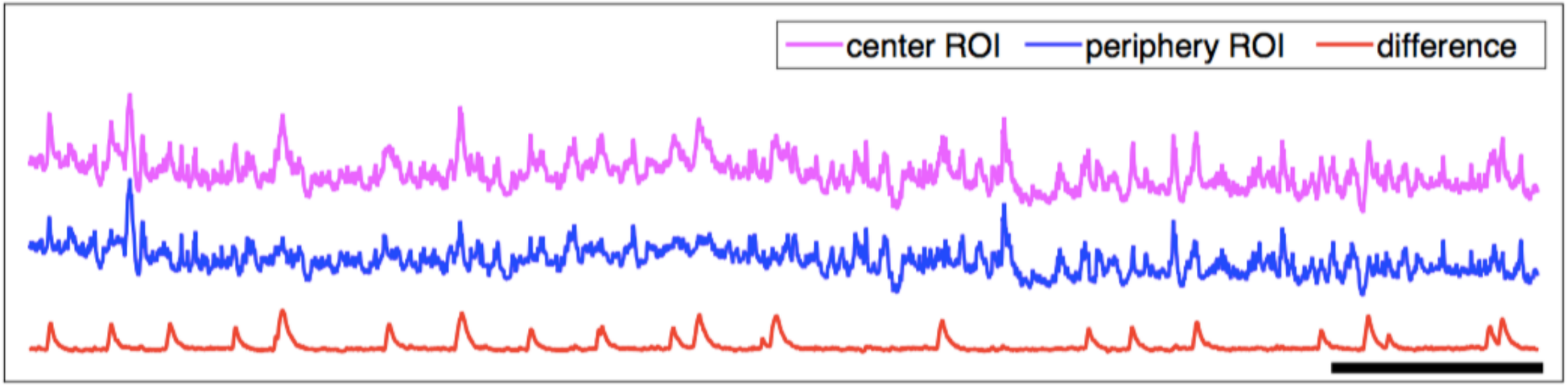




Pnevmatikakis et.al., arXiv 2014

## Spatial overlaps

# Problems

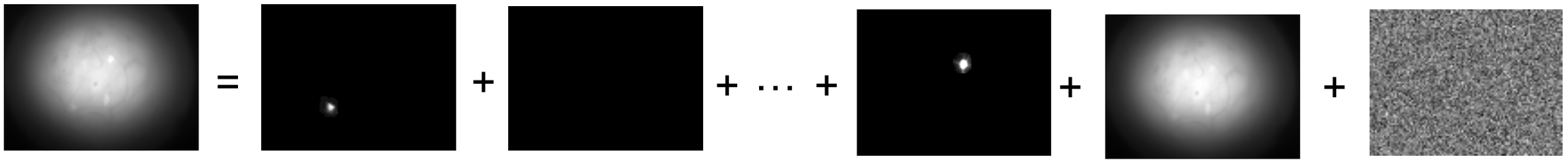




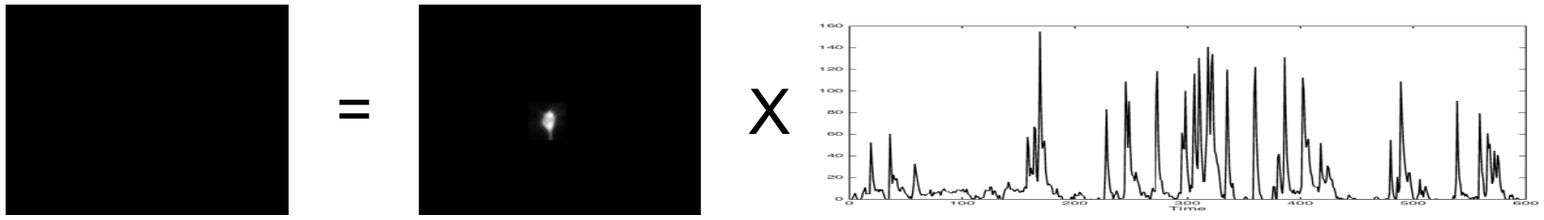

Zhou et.al., arXiv 2016

**extra fluorescence signals within ROI**

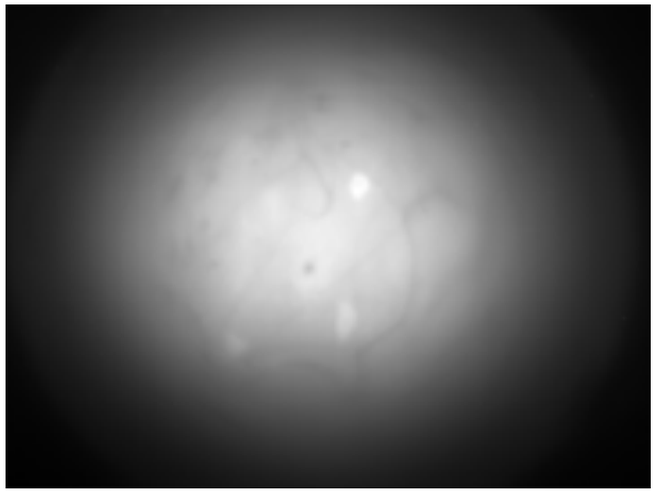
# Matrix Factorization Approach

1. 


data | neuron 1 | neuron 2 | neuron K | background | noise

2. 


spatial-temporal activity | neuron shape | temporal activity

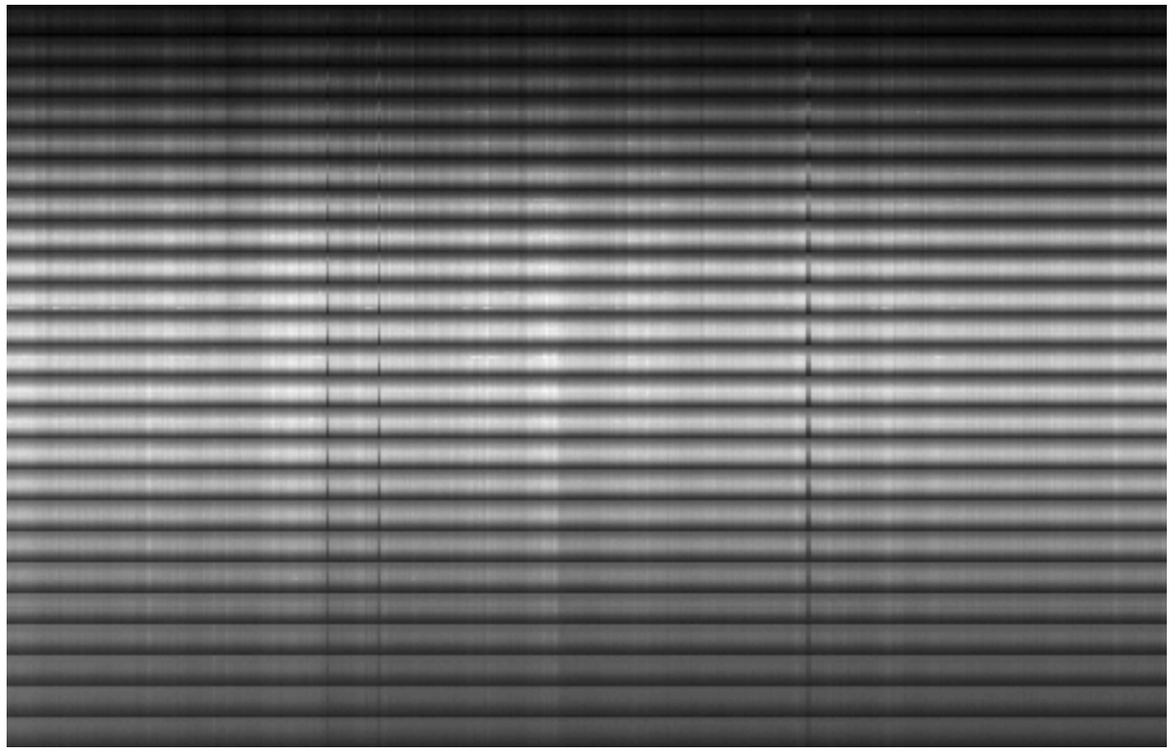


3D video

=

d pixel

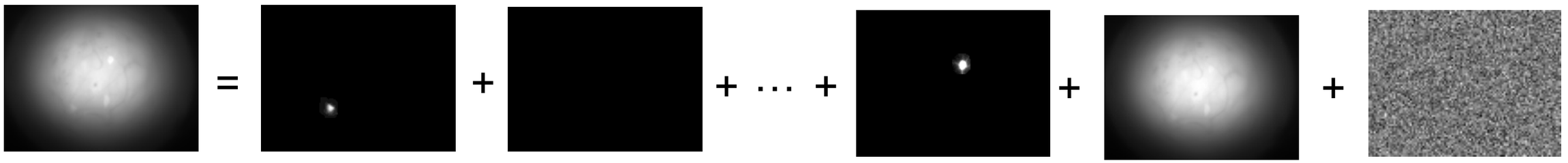
T frames



2D matrix

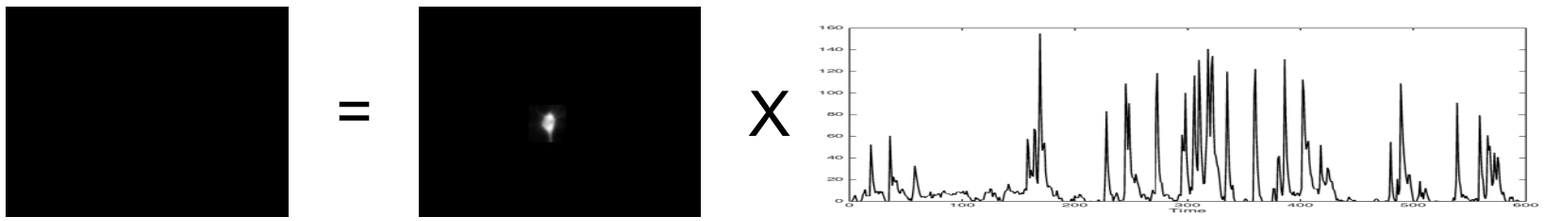
We represent a video data as a matrix $Y \in \mathbb{R}_{+}^{d \times T}$

# Matrix Factorization Approach

1. 


data | neuron 1 | neuron 2 | neuron K | background | noise

$$Y = Y_1 + Y_2 + \cdots + Y_K + B + E$$

2. 


spatial-temporal activity | neuron shape | temporal activity

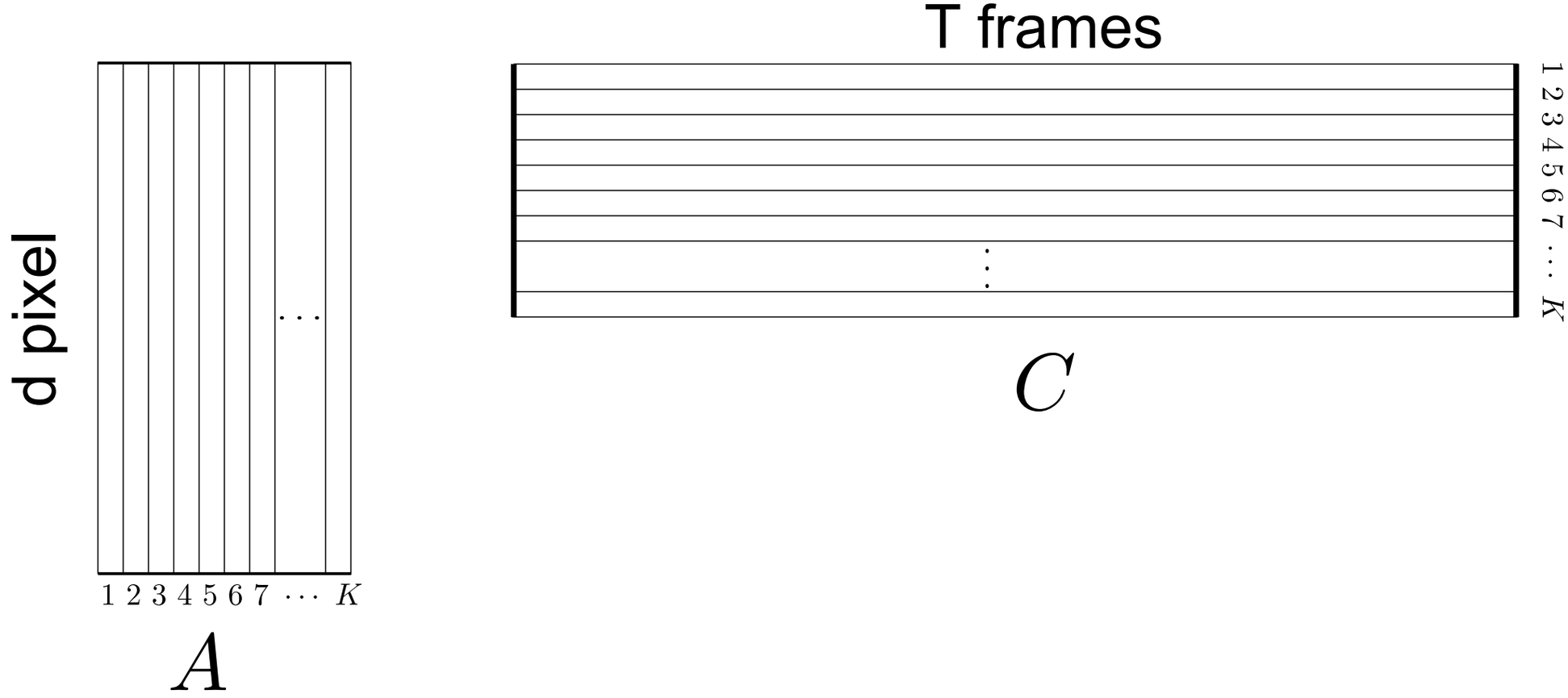
$$Y_i = a_i \cdot c_i^T, \; a_i \in \mathbb{R}^d \text{ and } c_i \in \mathbb{R}^T$$

# Matrix Factorization Approach

$$Y = Y_1 + Y_2 + \cdots + Y_K + B + E$$

$$Y_i = a_i \cdot c_i^T, \; a_i \in \mathbb{R}^d \text{ and } c_i \in \mathbb{R}^T$$

$$\Rightarrow Y = AC + B + E$$

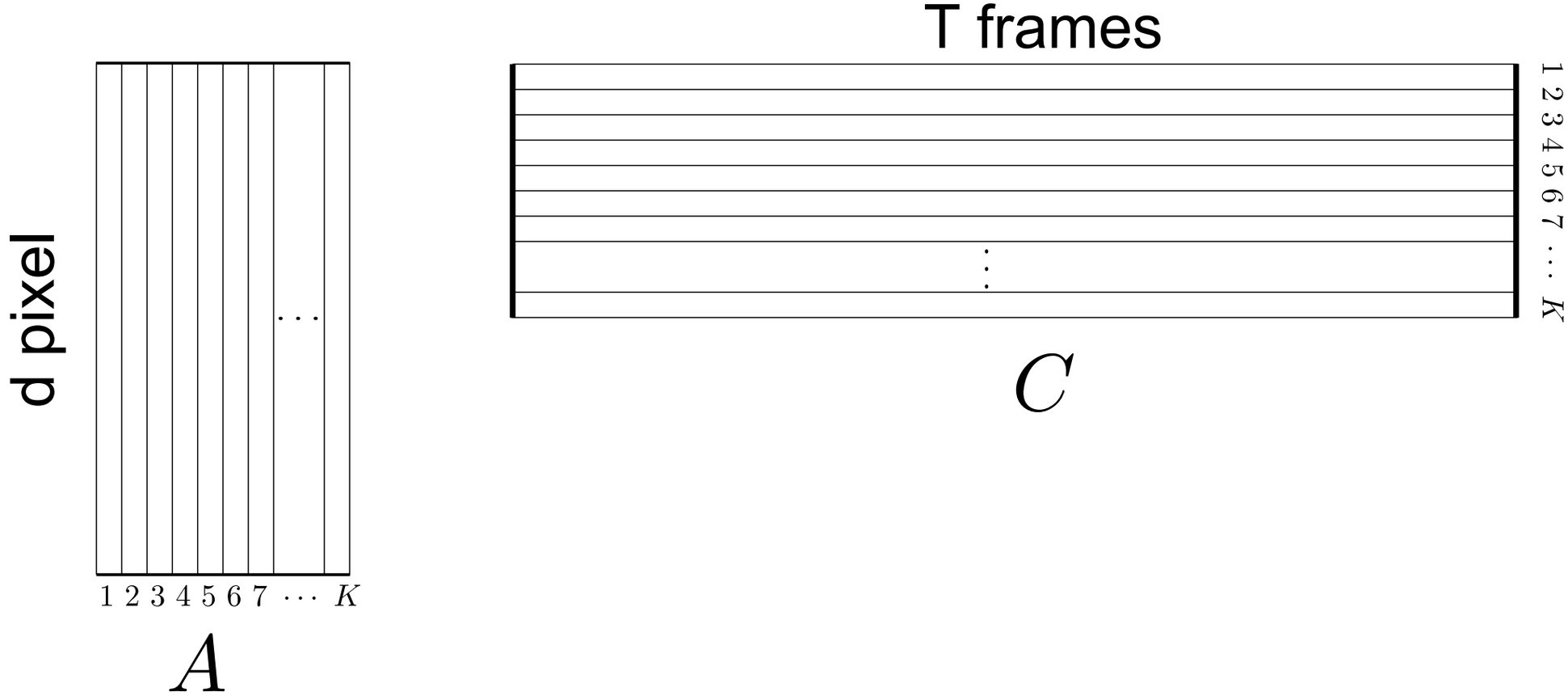


# Matrix Factorization Approach

$$Y = AC + B + E$$

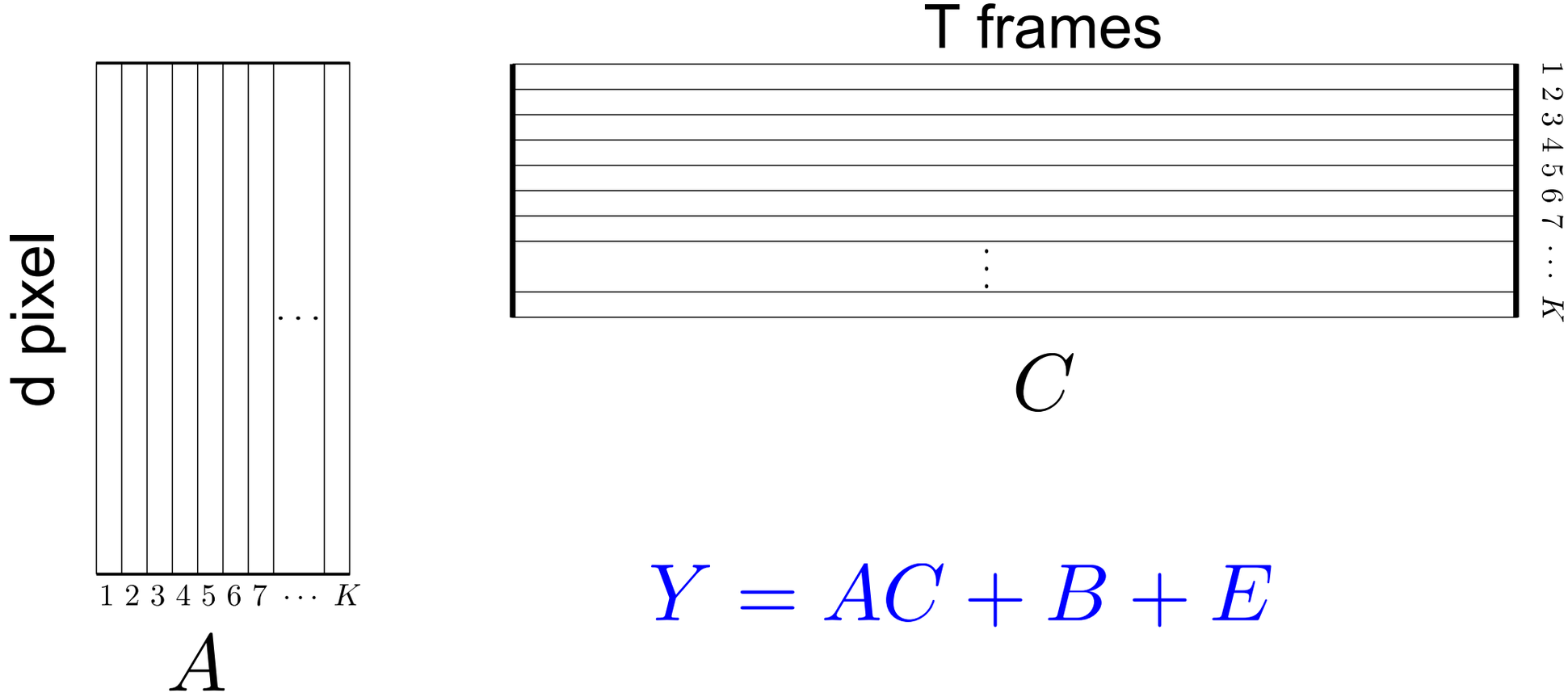
**Cell extraction problem can be solved by finding A & C.**

1. each column of A is the spatial shape of one neuron
2. each row of C is the temporal activity of one neuron



# Matrix Factorization Approach

- simultaneously identifying neurons and extracting their temporal traces.
- demixing spatially overlapped neural signals.




$$Y = AC + B + E$$

# Matrix Factorization Approach

$$Y = AC + B + E$$

1. Finding A & C is not easy.

2. It requires reasonable **constraints** to model variables.

3. The quality of cell extraction depends on
   - how **realistic** are added constraints
   - how well we can **estimate** A & C given these constraints.

# Matrix Factorization Approach

1. **PCA/ICA** (Mukamel et al., Neuron 2009)
   - Neuron shapes and temporal components are spatially and temporally independent.

2. **CNMF** (Pnevmatikakis et al., Neuron 2016)
   - Constrained Nonnegative Matrix Factorization (CNMF)
   - Simultaneous denoising, deconvolution, and demixing of Calcium Imaging Data
   - to be discussed

**Other formulations:**
SHMF (Diego & Hamprecht 2013);
NMF (Maruyama et al. 2014);
SSTD(Diego & Hamprecht 2014)

# CNMF framework

$$Y = \sum_{i=1}^{K} \mathbf{a}_i \cdot \mathbf{c}_i^T + B + E = AC + B + E$$

white noise

| Name | Description | Dimension |
|---|---|---|
| $Y$ | motion corrected video data | $\mathbb{R}_+^{d \times T}$ |
| $A$ | spatial footprints of all neurons | $\mathbb{R}_+^{d \times K}$ |
| $C$ | temporal activity of all neurons | $\mathbb{R}_+^{K \times T}$ |
| $B$ | background activity | $\mathbb{R}_+^{d \times T}$ |
| $E$ | observation noise | $\mathbb{R}^{d \times T}$ |
| $d$ | number of pixels | integer |
| $T$ | number of frames | integer |
| $K$ | number of neurons | integer |

Pnevmatikakis et al., Neuron 2016

# CNMF framework

- Both the spatial and the temporal components are nonnegative

$$A \geq 0,\ C \geq 0$$

- The spatial components have localized and compact footprints

$$A \text{ is sparse}$$

- The temporal activity of each neuron obeys the dynamics of calcium indicators

spiking signal

$$c_i(t) = \sum_{j=1}^{p} \gamma_j^{(i)} c_i(t-j) + s_i(t) \quad \Leftrightarrow \quad \mathbf{s}_i = G^{(i)} \mathbf{c}_i$$

- The spiking signals are sparse

$$\mathbf{s}_i \text{ is sparse}$$

Pnevmatikakis et al., Neuron 2016

# CNMF framework

- It is a **general** framework for analyzing calcium imaging data. It provides a generative model of data.
- Implementations to different datasets may require **extra** constraints to model variables.
- Estimating A & C is a nontrivial optimization problem.
- Finding a reasonable solution needs a good initialization method.

See Pnevmatikakis et al., Neuron 2016 for successful applications of CNMF.

- Part I: Overview of methods for source extraction
  - General methods for calcium imaging data
  - **Constrained Nonnegative Matrix Factorization for microEndoscopic data (CNMF-E)**
- Part II: Exercises

# Microendoscope


1. Ghosh, K. K., Burns, L. D., Cocker, E. D., Nimmerjahn, A., Ziv, Y., El Gamal, A., & Schnitzer, M. J. (2011). Miniaturized integration of a fluorescence microscope. Nature methods, 8(10), 871-878.
2. https://www.youtube.com/watch?v=K6Iw6VESQEo

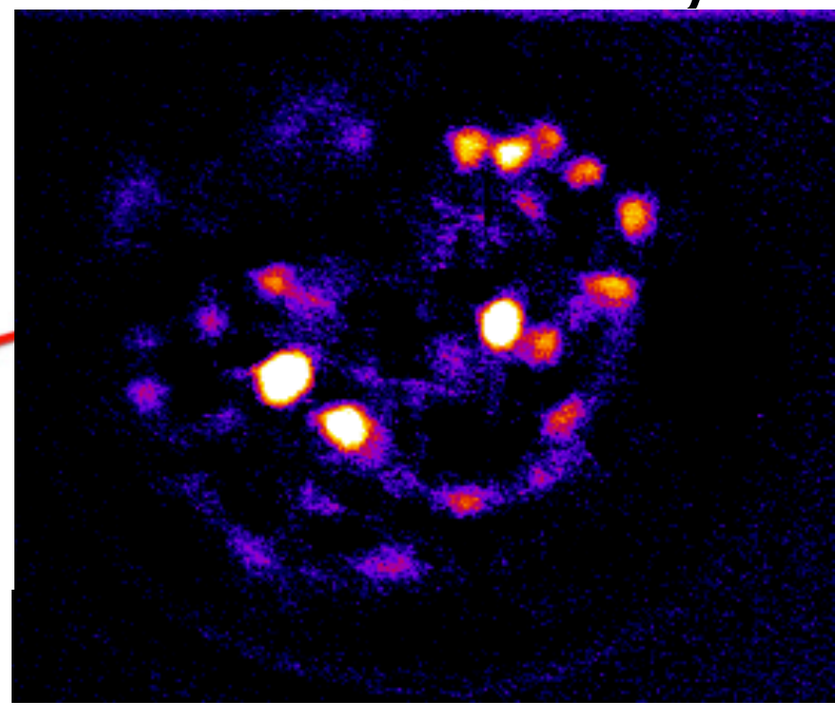
behavior





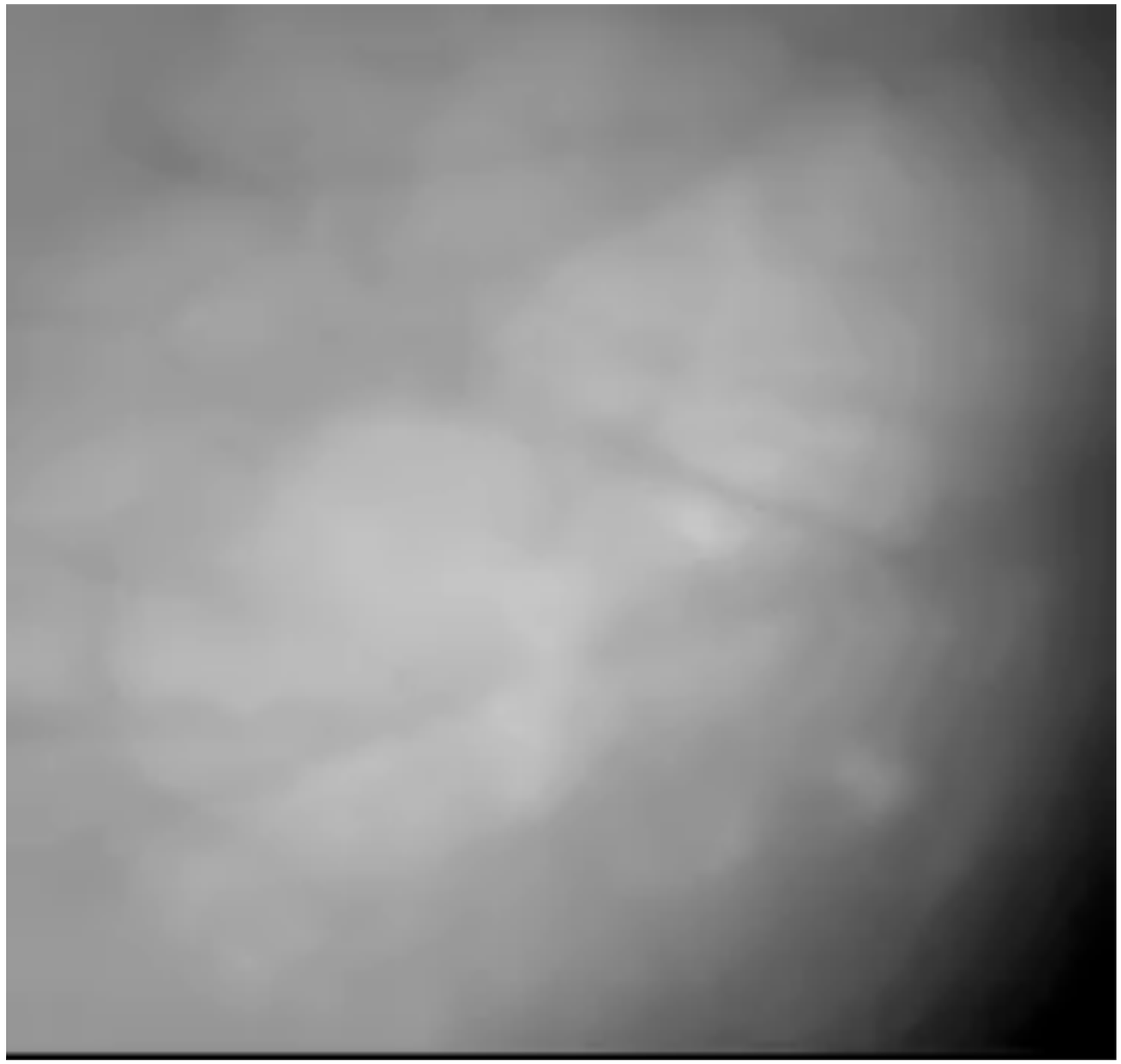
**Neuroscientists**

neural activity



# Microendoscopic Data

- strong background with rapid fluctuations
- many severely overlapped neurons
- low signal-to-noise ratios (SNRs)

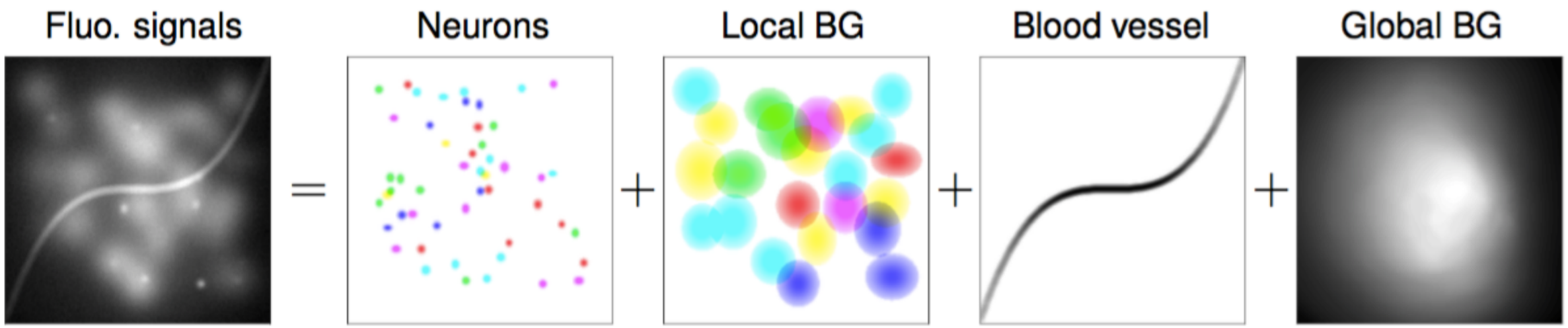


1. GCaMP6s labeled PFC neuron.
2. The data is recorded by Shanna Resendez in Garret Stuber's lab in UNC.

# CNMF-E

Constrained Nonnegative Matrix Factorization for microEndoscopic data (E also stands for extension).

$$Y = AC + B + E$$




1. BG: background

# CNMF-E

Compared  to the plain CNMF, CNMF-E contains following modifications

- A new background model
- A new algorithm for fitting model variables
- A new initialization method

# Model of the background

$$Y = AC + B + E$$

Precise estimation of the background is crucial for accurate extraction of neural signals.

CNMF

$$B = b \cdot f^T$$




- Microendoscopic data has multiple sources of the background fluctuations (high rank).
- Simply increasing the rank of the B may include neural signals into the background.

# Model of the background

## CNMF-E

$$B = B^f + \mathbf{b}_0 \cdot \mathbf{1}^T$$ fluctuating background + constant baselines

$$B^f \cdot \mathbf{1} = \mathbf{0}$$

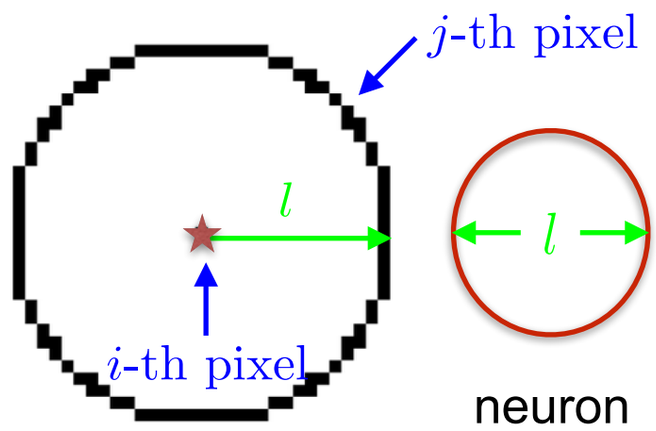
$$B^f = W B^f$$

$$W_{ij} = 0 \text{ if } \mathrm{dist}(i,j) \notin (l, l+1]$$

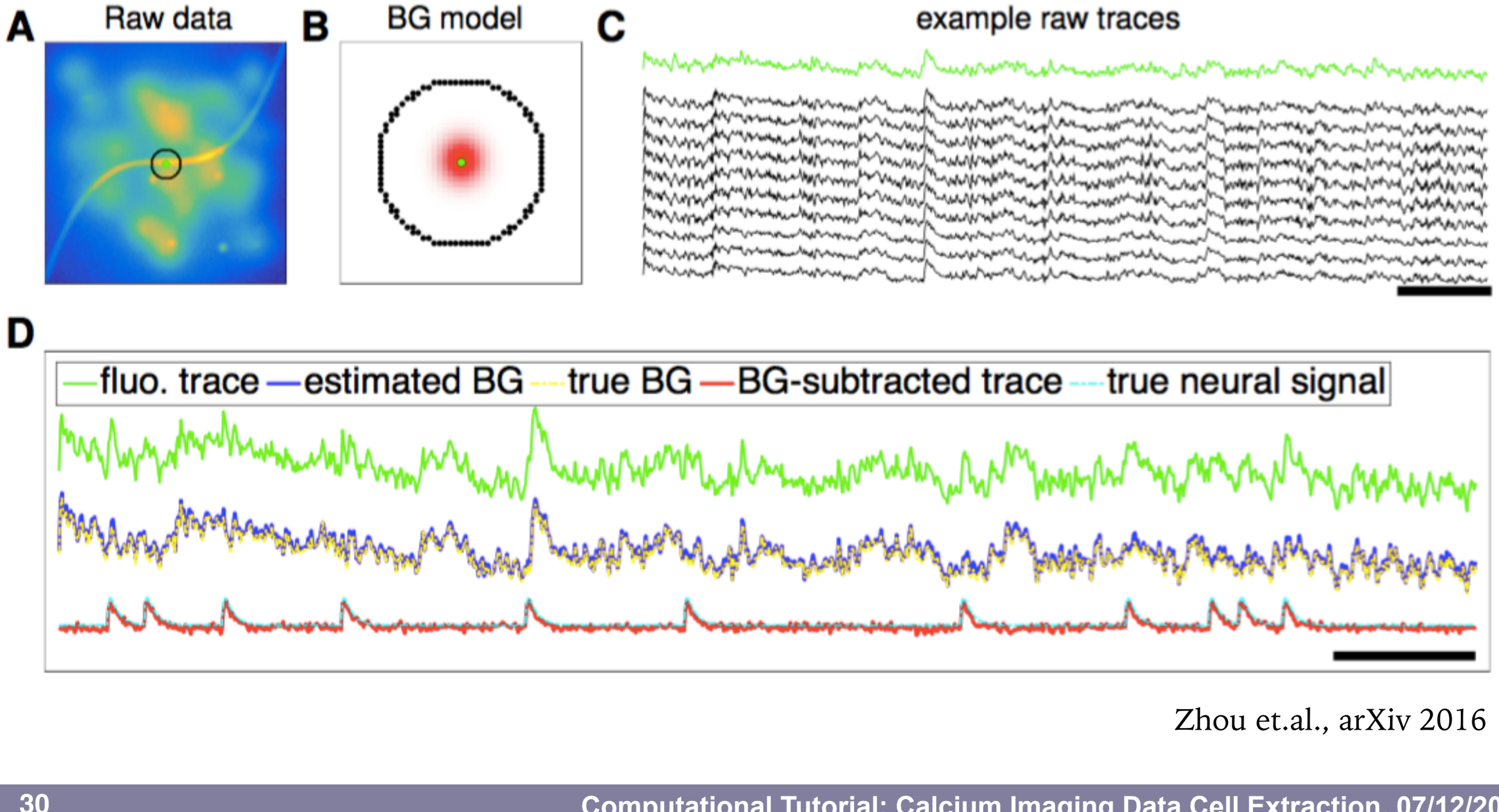
The background fluctuation at each pixel can be represented as a linear combination of its neighboring pixels' fluctuations.

$$B^f_{it} = \sum_{j \in \Omega_i} w_{ij} \cdot B^f_{jt}, \quad \forall t = 1 \ldots T$$

$$\Omega_i = \{j | \mathrm{dist}(i,j) \in (l, l+1]\}$$

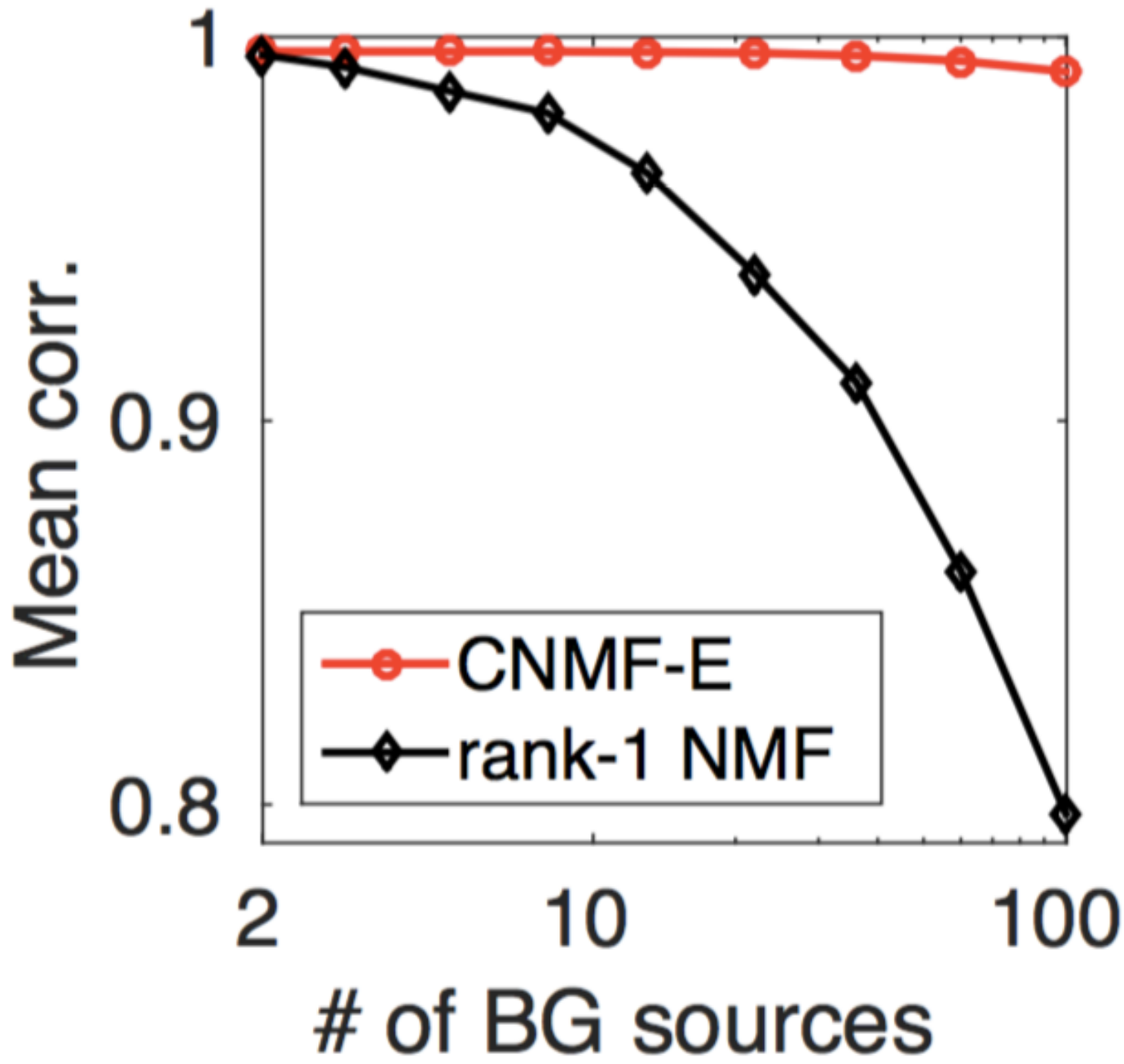


# The background model in CNMF-E can accurately recover the true background  fluctuations in simulated data.




Zhou et.al., arXiv 2016

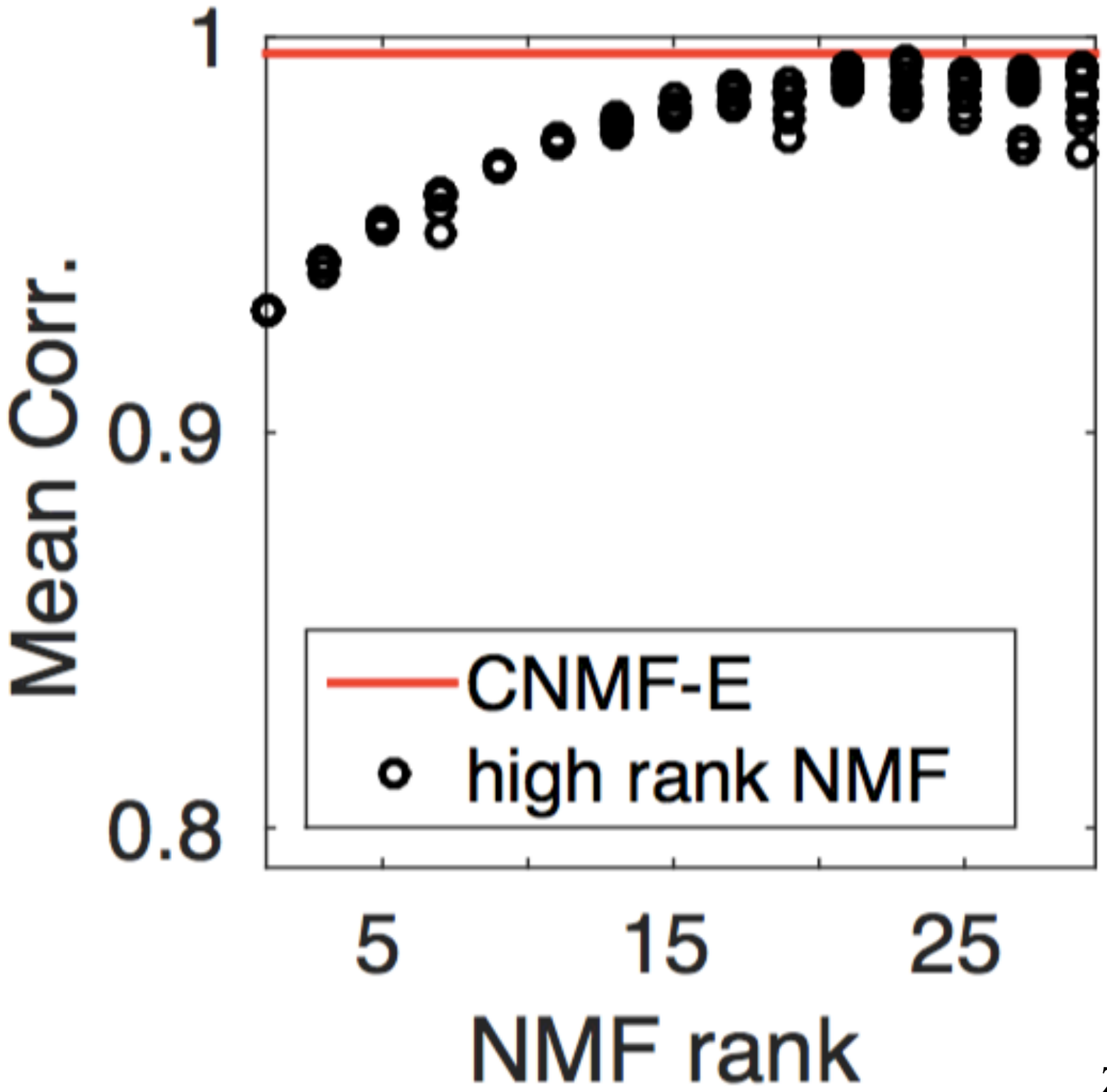
When number of background (BG) sources increases, CNMF-E can still estimate the background component B with  high accuracy.




Zhou et.al., arXiv 2016

1. Mean corr. : mean correlation coefficient between the true background fluctuation and the estimated background fluctuation at each pixel.

The results of higher rank NMF model are still inferior to CNMF-E.




Zhou et.al., arXiv 2016

# Solving CNMF-E

Summary of the model

$$Y = AC + \boxed{B^f + \mathbf{b}_0 \cdot \mathbf{1}^T} + E$$

$$A \geq 0, A \text{ is sparse}$$

$$\mathbf{c}_i \geq 0,\ \mathbf{s}_i \geq 0,\ G^{(i)}\mathbf{c}_i = \mathbf{s}_i,\ \mathbf{s}_i \text{ is sparse}$$

} neuron

$$B^f \cdot \mathbf{1} = \mathbf{0}$$

$$B^f = W \cdot B^f$$

$$W_{ij} = 0 \text{ if } \text{dist}(i,j) \notin (l, l+1]$$

} background

# Optimization problem

$$\min_{A,C,W,\mathbf{b}_0} \quad \|Y - AC - \mathbf{b}_0 \cdot \mathbf{1}^T - B^f\|_F^2$$

$$\text{s.t.} \quad A \geq 0, A \text{ is sparse}$$

$$\mathbf{c}_i \geq 0, \mathbf{s}_i \geq 0, G^{(i)}\mathbf{c}_i = \mathbf{s}_i, \mathbf{s}_i \text{ is sparse} \quad \forall i = 1 \dots K$$

$$B^f \cdot \mathbf{1} = \mathbf{0}$$

$$B^f = W \cdot (Y - AC - \mathbf{b}_0 \cdot \mathbf{1}^T)$$

**non-convex**

$$W_{ij} = 0 \text{ if } \text{dist}(i,j) \notin (l, l+1]$$

$\hat{A} \in \mathbb{R}_+^{d \times K}$ gives the spatial components of $K$ neurons

$\hat{C} \in \mathbb{R}_+^{K \times T}$ gives the temporal components of $K$ neurons

The fluctuating background is $B^f = \hat{W}(Y - \hat{A}\hat{C} - \hat{\mathbf{b}}_0 \cdot \mathbf{1}^T)$

1. For $X \in \mathbb{R}^{d \times T}$, its Frobenius norm is defined as $\|X\|_F^2 = \sum_{i=1}^{d}\sum_{j=1}^{T} X_{ij}^2$

# Three subproblems

**Estimating $A$ given $\hat{C}, \hat{B}^f, \hat{\boldsymbol{b}}_0$:**

$$\min_{A} \quad \|Y - A \cdot \hat{C} - \hat{\boldsymbol{b}}_0 \cdot \mathbf{1}^T - \hat{B}^f\|_F^2 \qquad \text{(P-S)}$$
$$\text{s.t.} \quad A \geq 0 \text{ and } A \text{ is sparse.}$$

**Estimating $C$ given $\hat{B}^f, \hat{\boldsymbol{b}}_0, \hat{A}$:**

$$\min_{C} \quad \|Y - \hat{A} \cdot C - \hat{\boldsymbol{b}}_0 \cdot \mathbf{1}^T - \hat{B}^f\|_F^2 \qquad \text{(P-T)}$$
$$\text{s.t.} \quad \boldsymbol{c}_i \geq 0, \boldsymbol{s}_i \geq 0$$
$$G^{(i)}\boldsymbol{c}_i = \boldsymbol{s}_i, \boldsymbol{s}_i \text{ is sparse} \quad \forall i = 1 \ldots K$$

**Estimating $B^f, \boldsymbol{b}_0$ given $\hat{A}, \hat{C}$**

$$\min_{W, \boldsymbol{b}_0} \quad \|Y - \hat{A} \cdot \hat{C} - \boldsymbol{b}_0 \cdot \mathbf{1}^T - B^f\|_F^2 \qquad \text{(P-B)}$$
$$\text{s.t.} \quad B^f \cdot \mathbf{1} = \mathbf{0}$$
$$B^f = W \cdot (Y - \hat{A} \cdot \hat{C} - \boldsymbol{b}_0 \cdot \mathbf{1}^T).$$
$$W_{ij} = 0 \quad \text{if dist}(i, j) \notin (l, l+1]$$

Zhou et.al., arXiv 2016

# Three subproblems

This strategy gives us the flexibility of including further potential interventions (either automatic or semi-manual) in the optimization procedure.

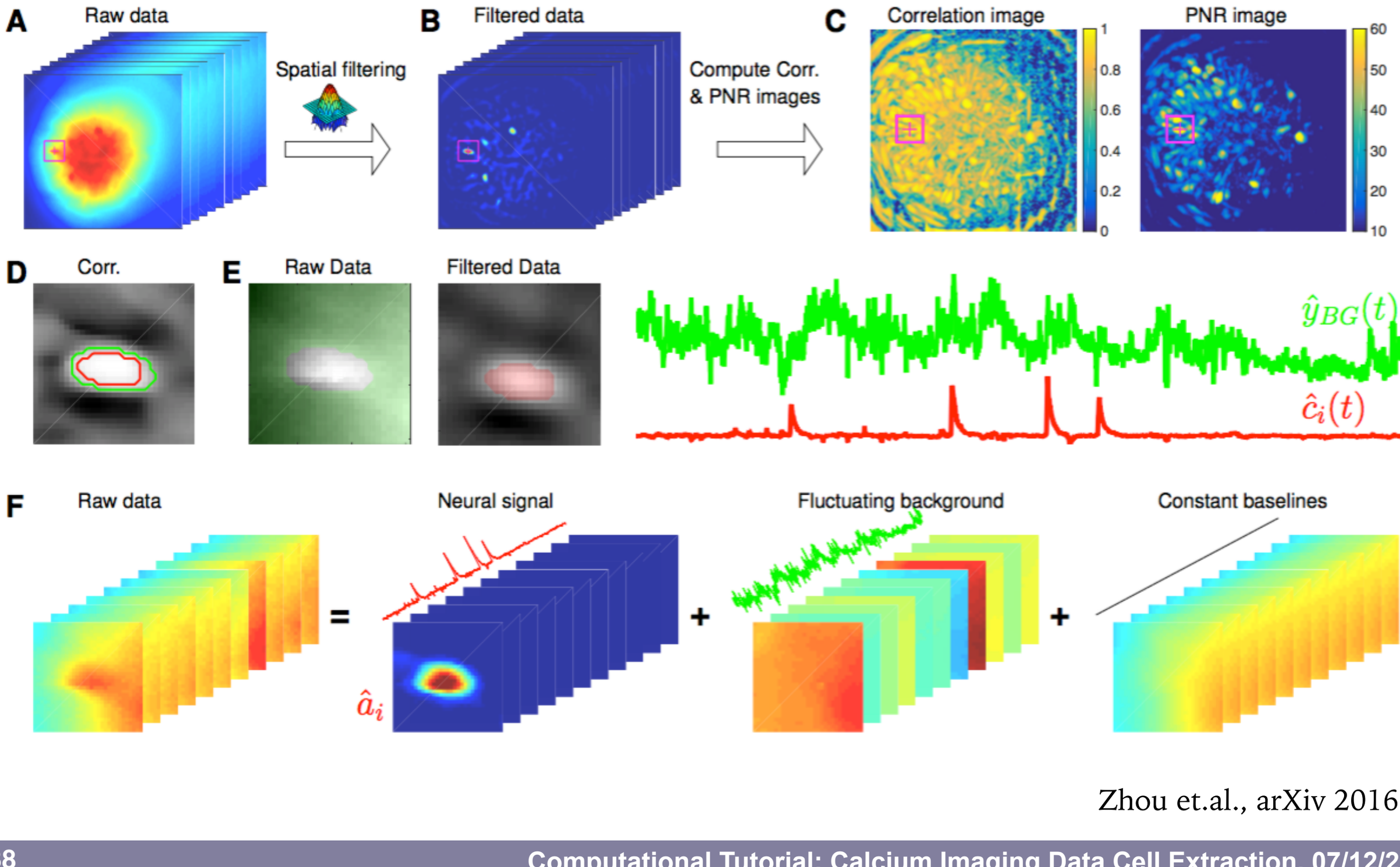
Zhou et.al., arXiv 2016

# Interventions

CNMF-E allows many interventions during the model fitting. More iterations following the interventions produces better results.

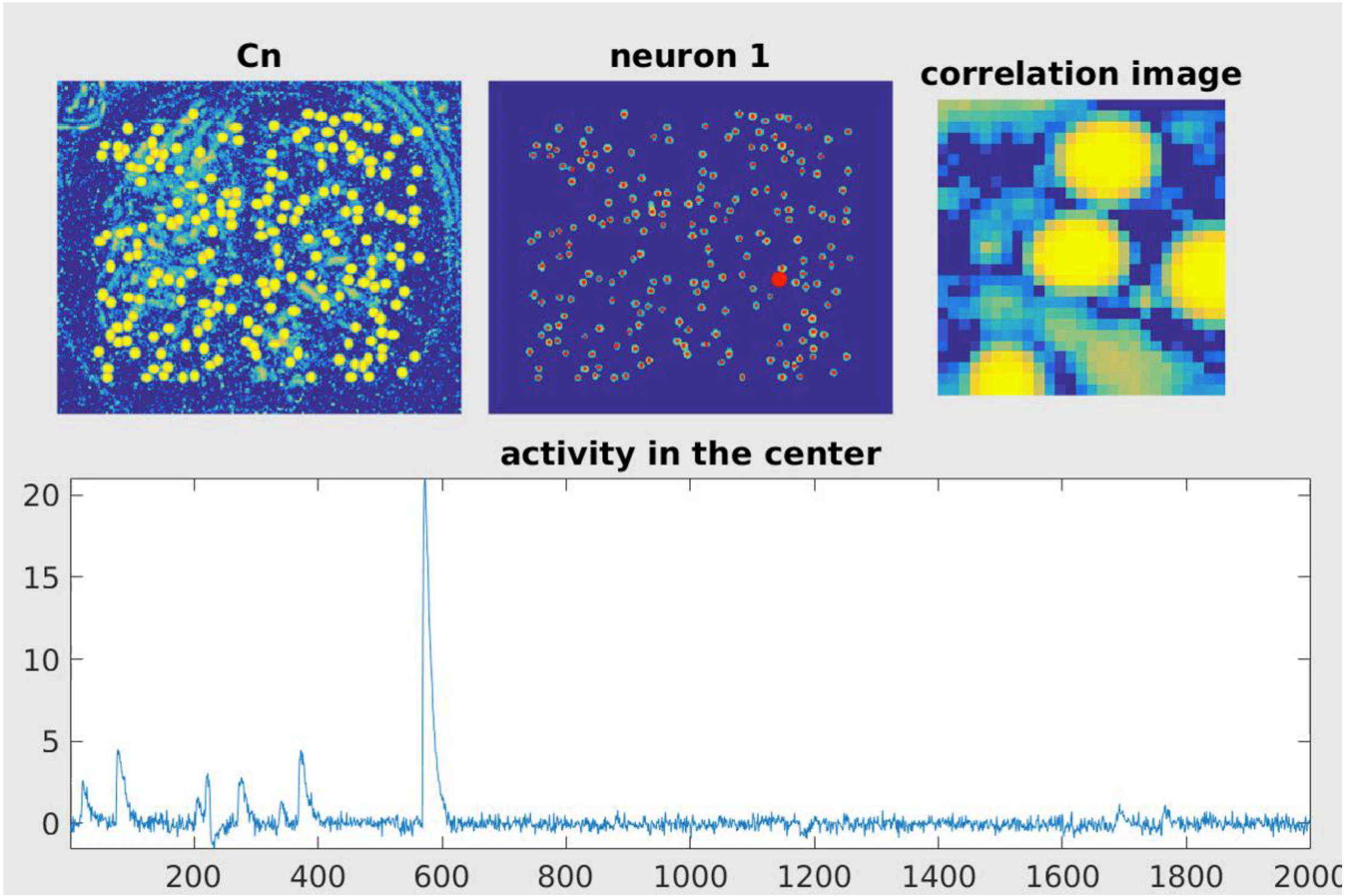
- merge components into one neuron (automatically)
  - merge neurons with high temporal correlation
  - merge neurons whose centers are too close
- split a component into multiple neurons
- remove false positive components
- pick neurons from residuals (automatically)

# Initialization

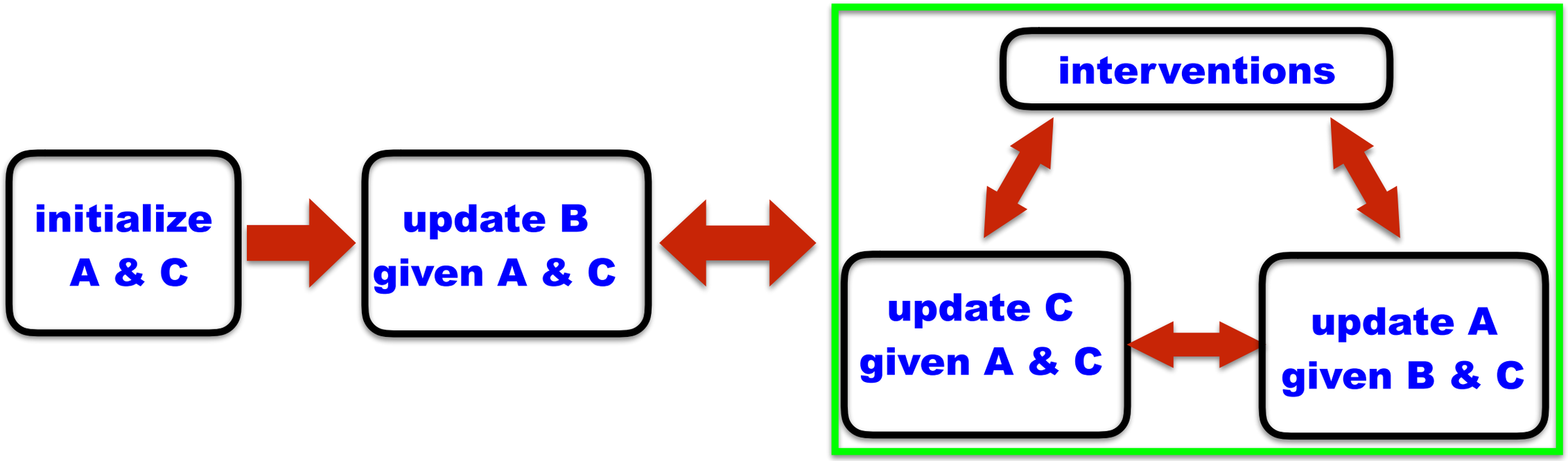



Zhou et.al., arXiv 2016

# Example


Cn
neuron 1
correlation image
activity in the center
20
15
10
5
0
200
400
600
800
1000
1200
1400
1600
1800
2000

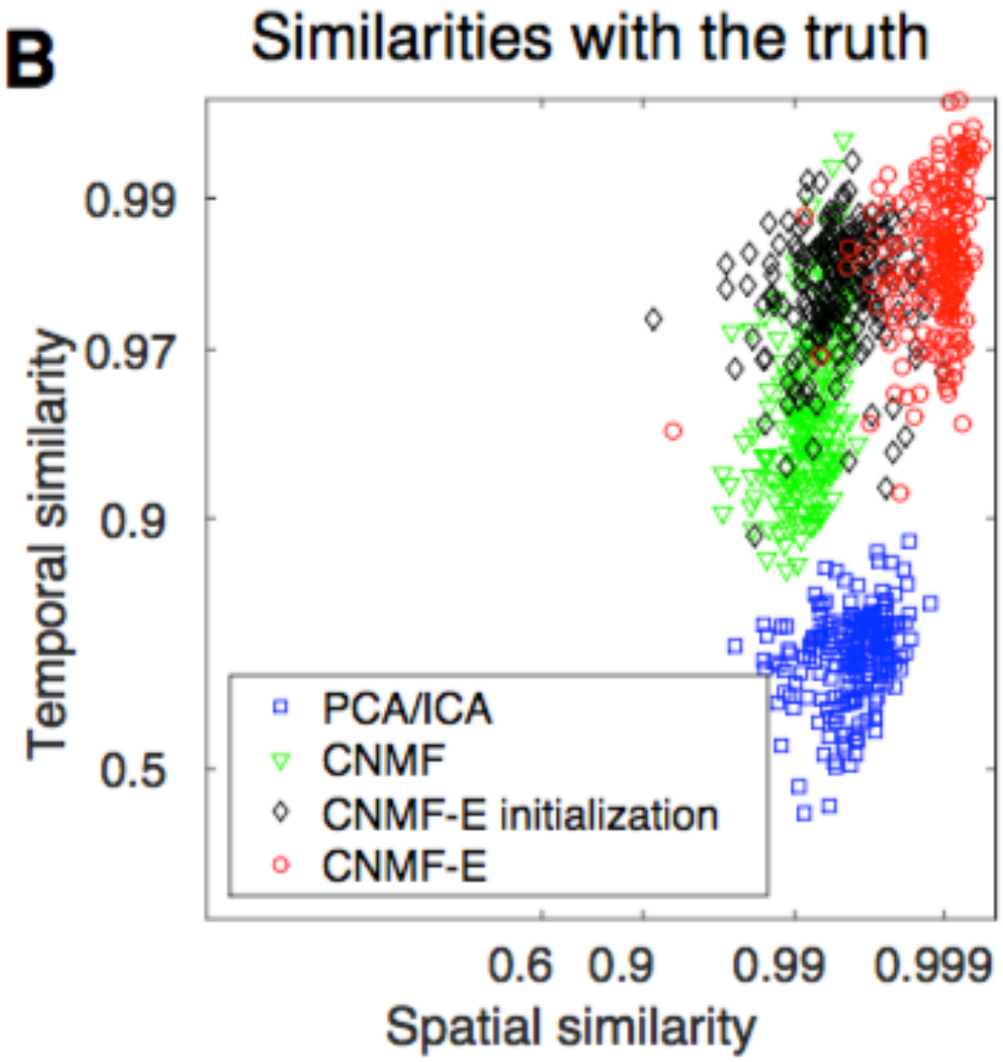
# Pipeline

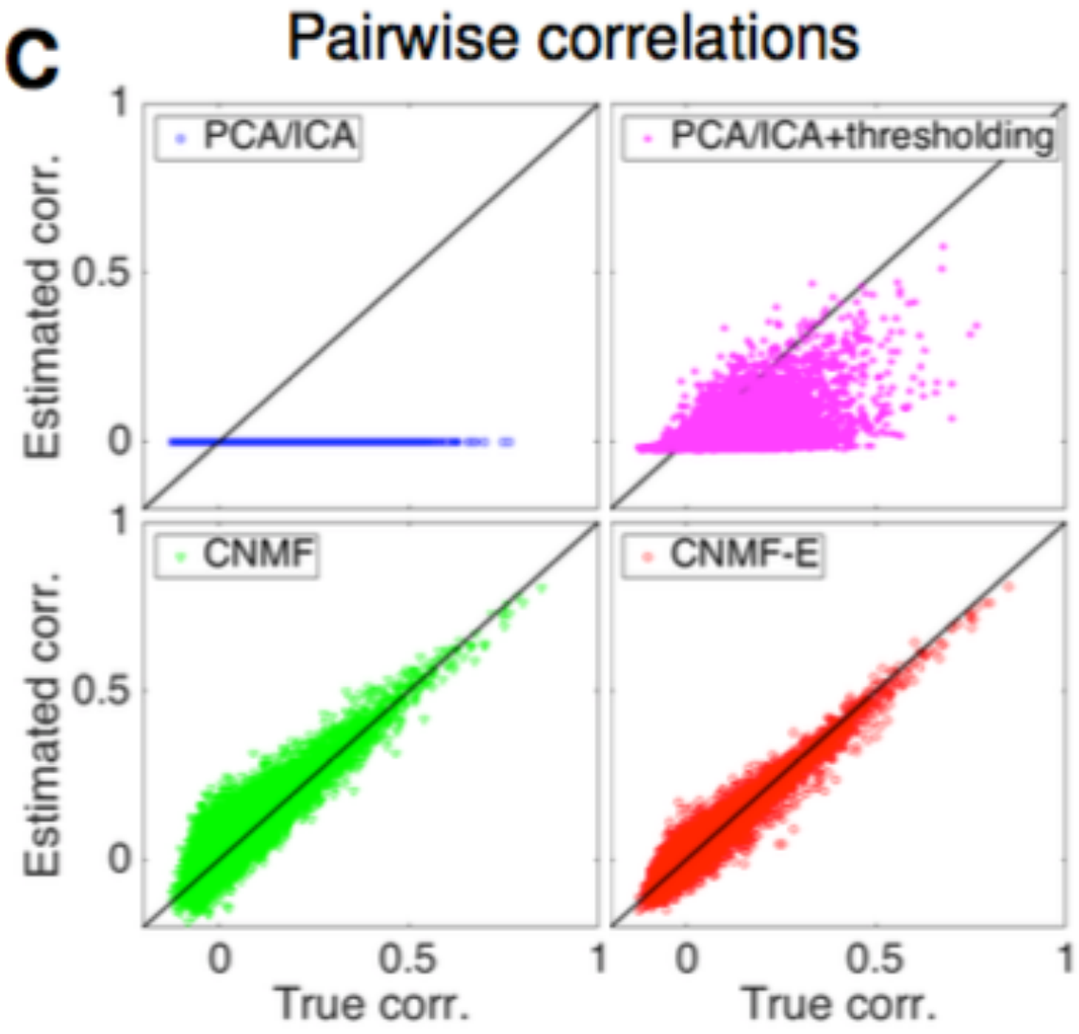



1. A: spatial filter of all neurons
2. B: background activity
3. C: temporal activity of all neurons

# Results 1: Simulation

CNMF-E recovers neural signals with high accuracy;
CNMF-E preserves the pair-wise correlation between neurons.

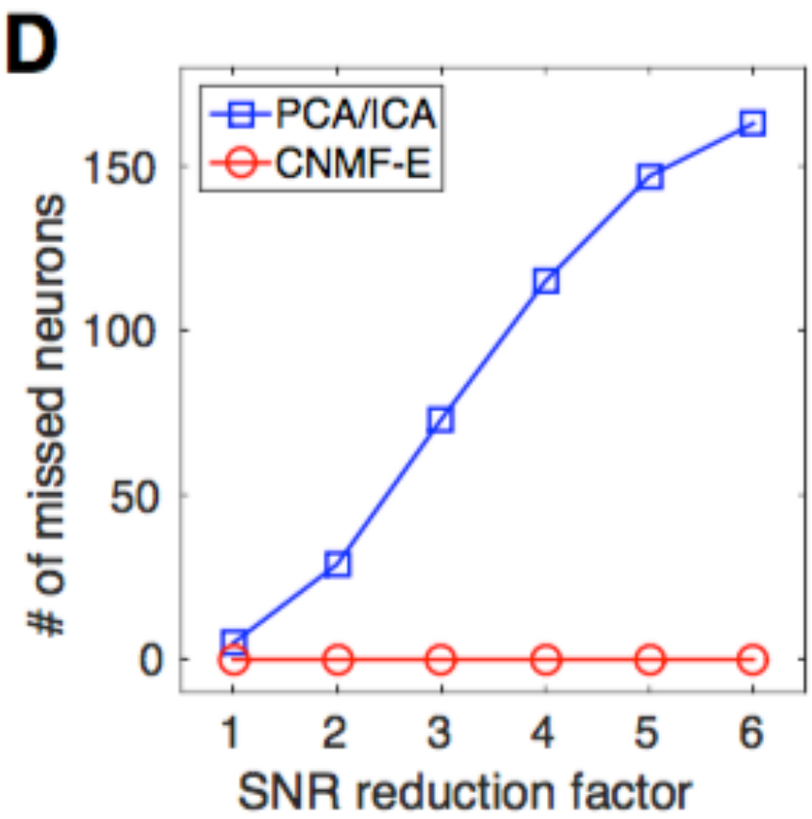


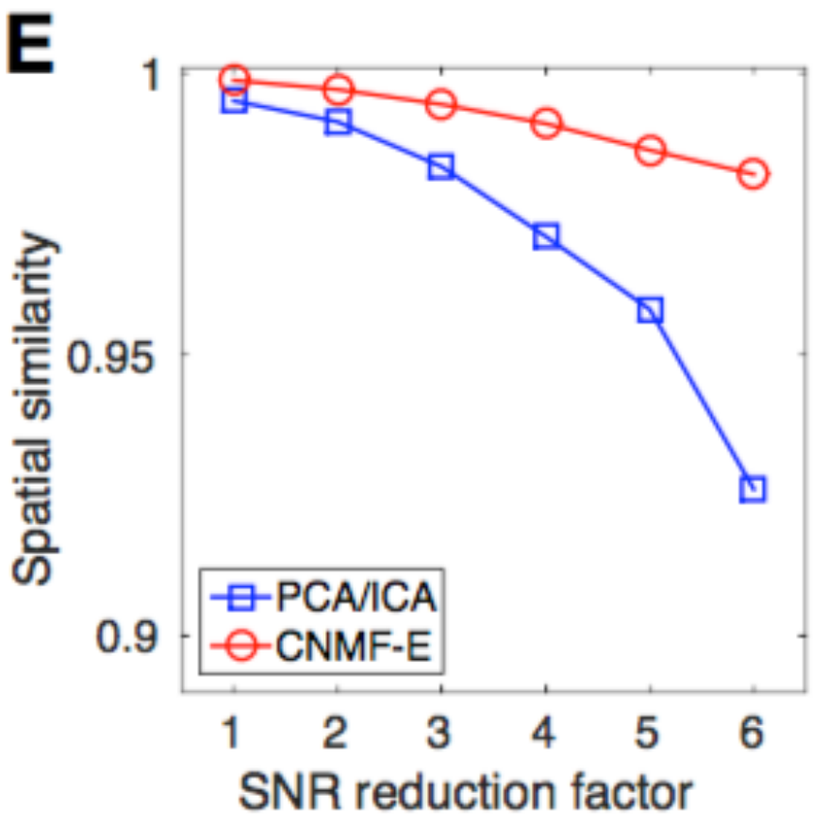



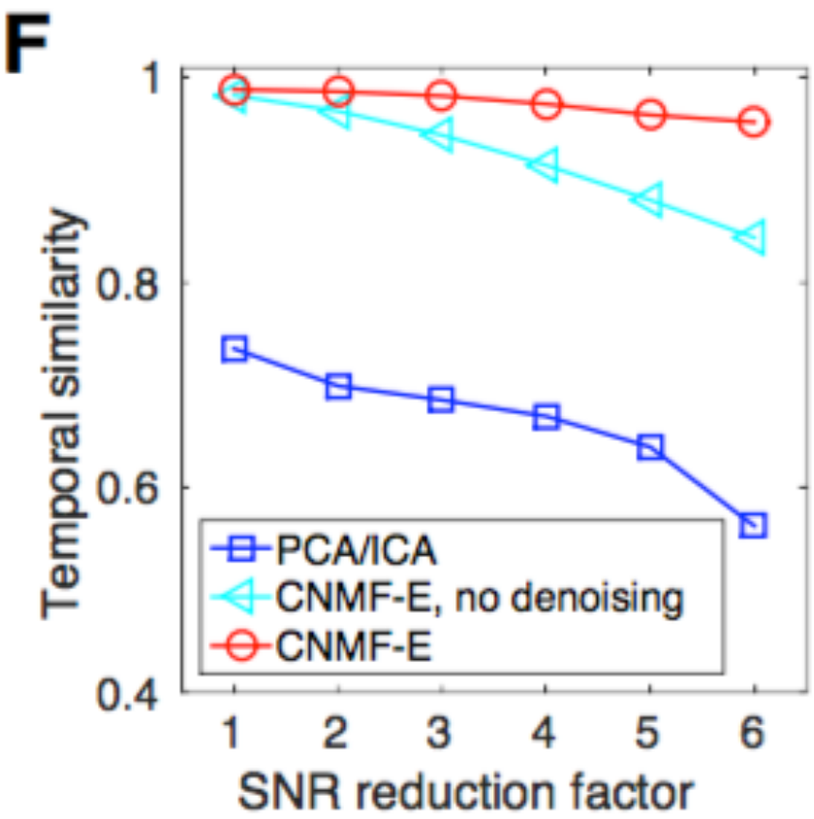

1. We use the ground truth as initialization of CNMF method.

# Example 1: Simulation

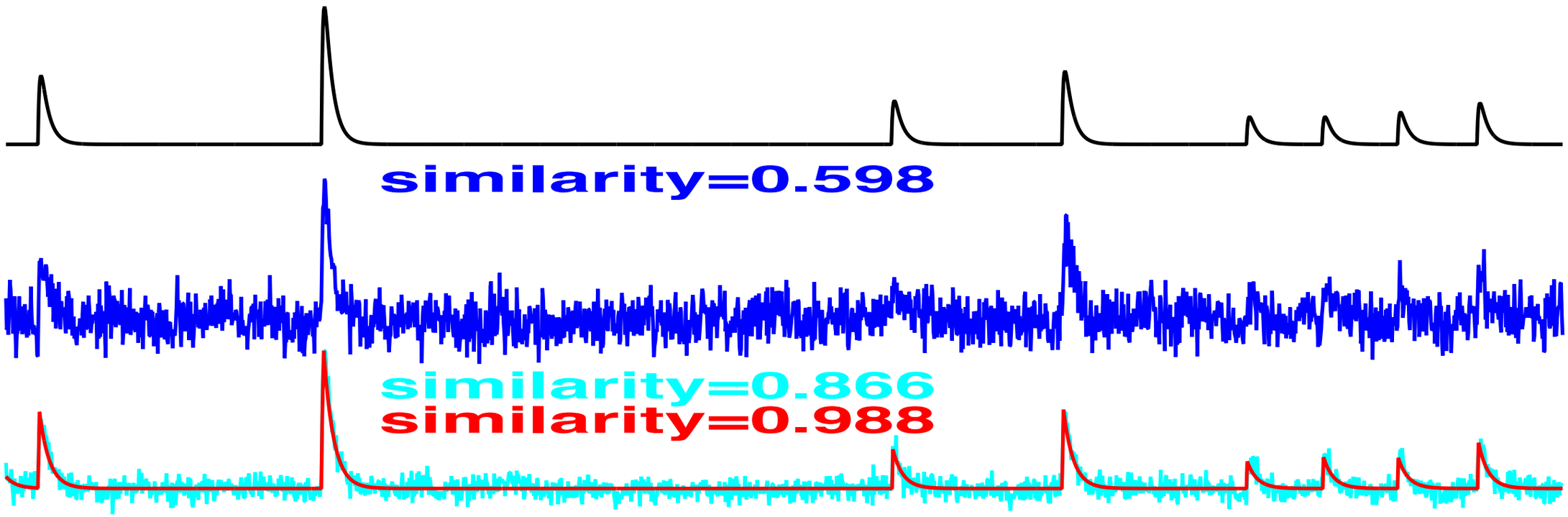
CNMF-E is robust to noise compared to PCA/ICA analysis.



# Example 1: Simulation




ground truth (**black**);

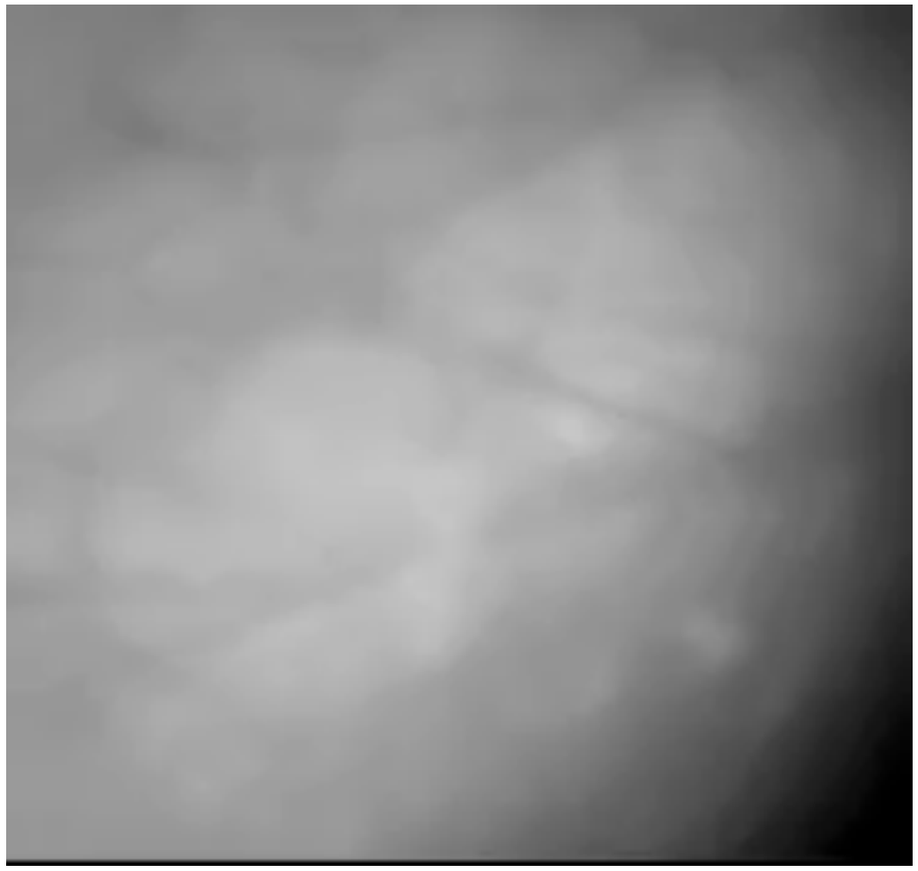
PCA/ICA trace (**blue**);

CNMF-E trace (**red**);

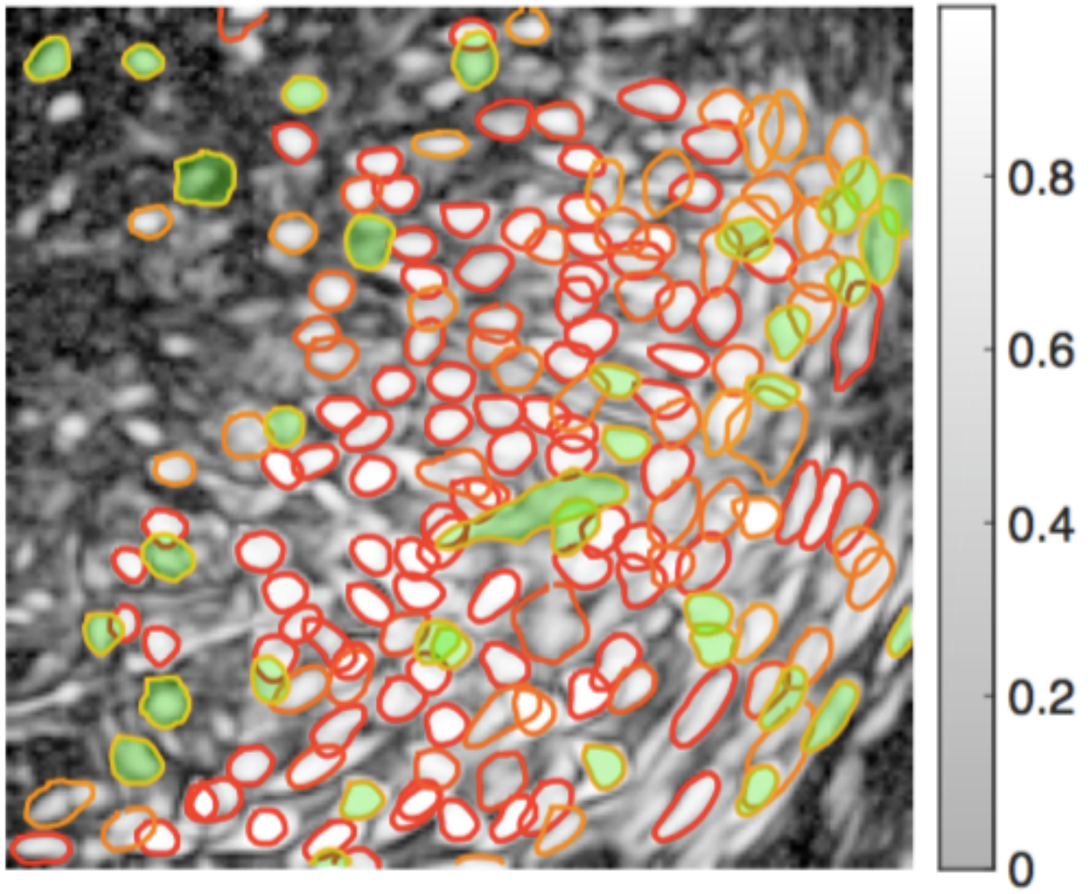
CNMF-E trace without being denoised (**cyan**)

# Example 2: in vivo PFC data

Raw data

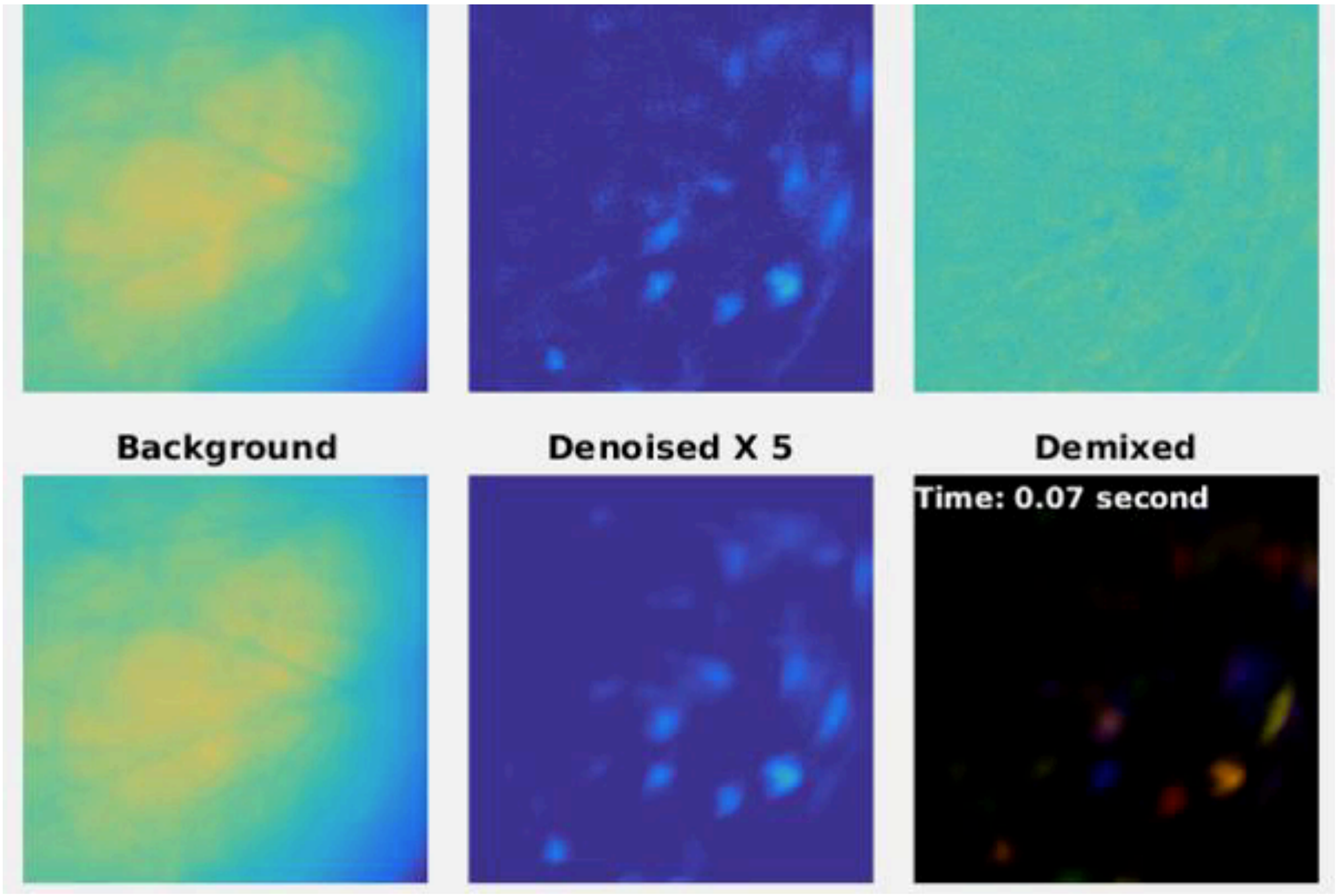


Identified neurons



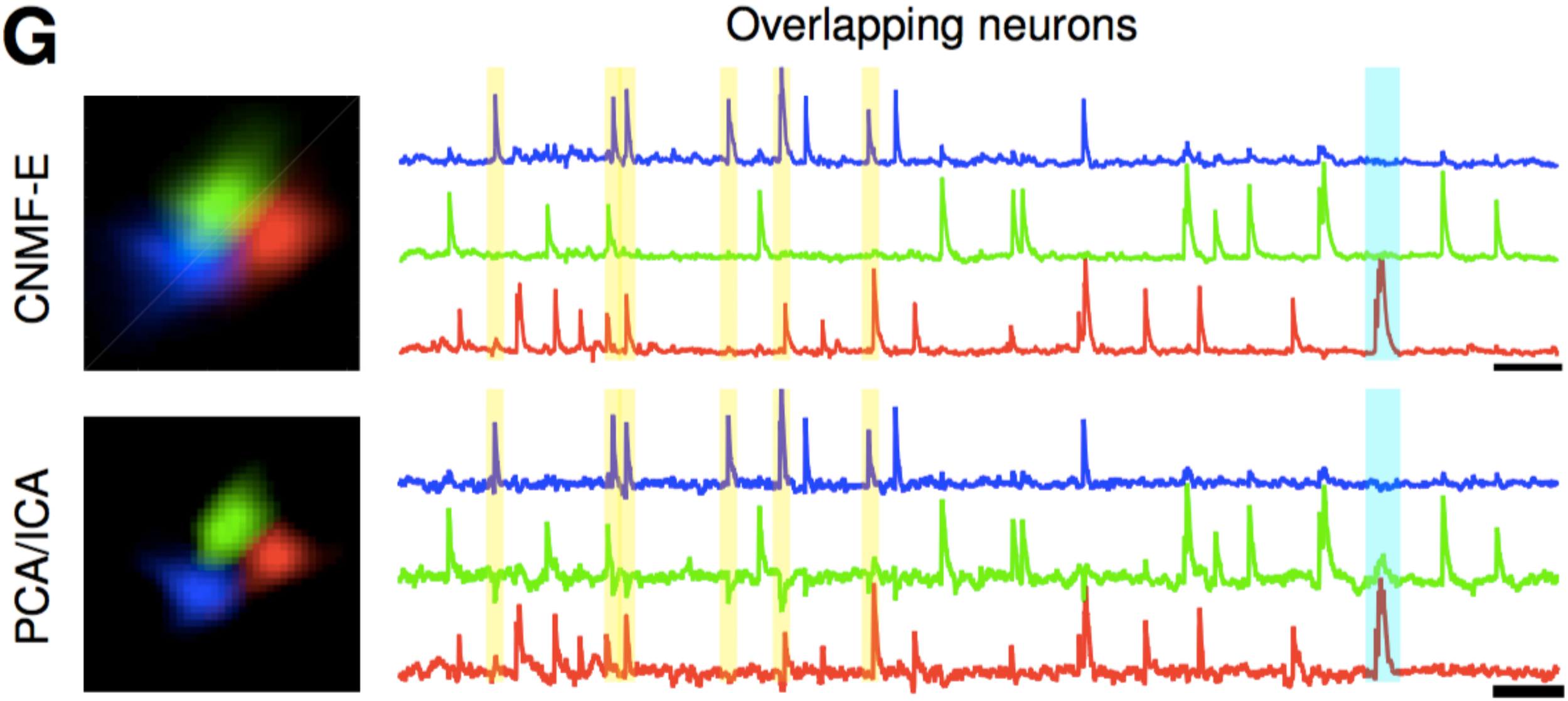

Zhou et.al., arXiv 2016

# Example 2: in vivo PFC data



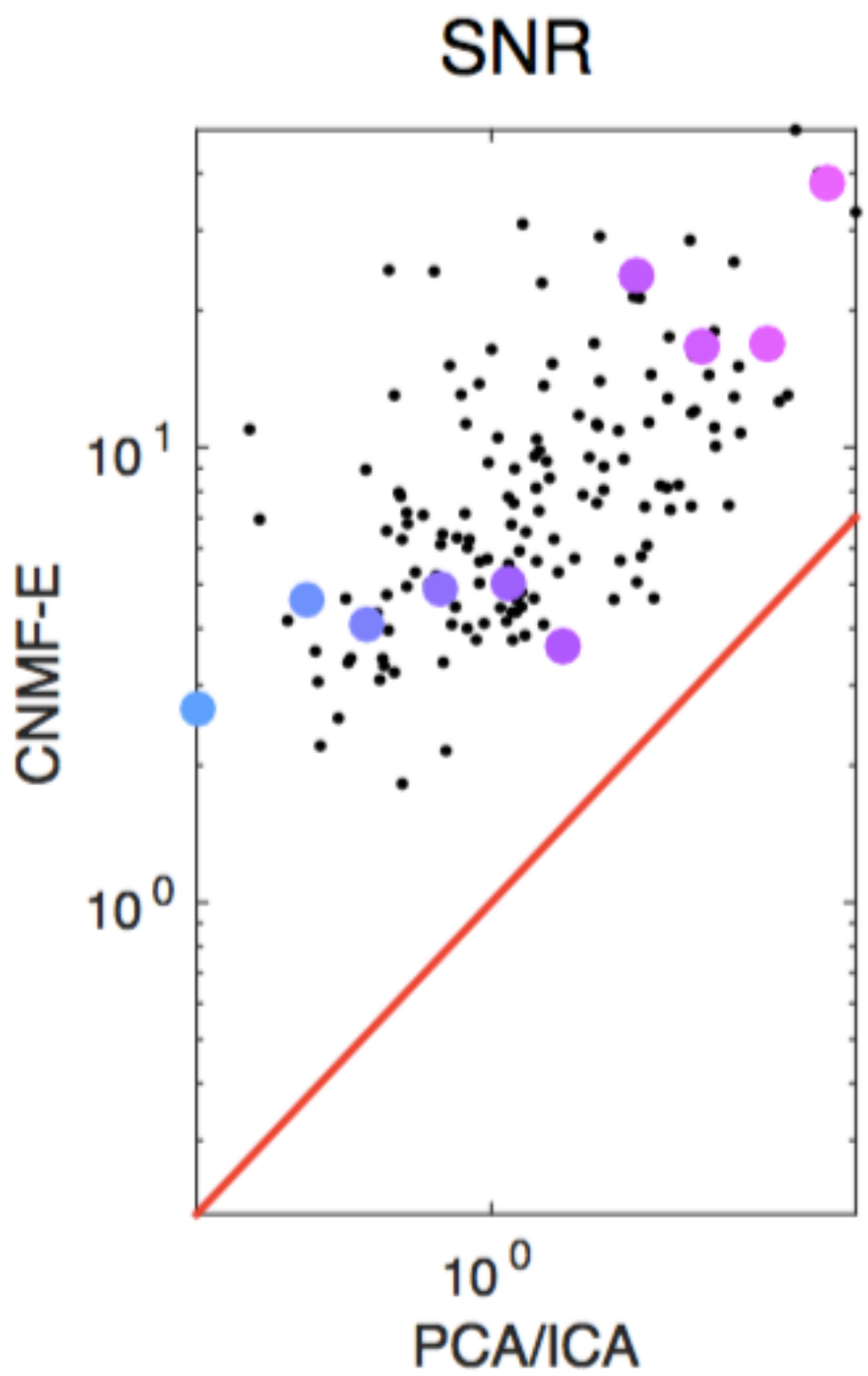

Zhou et.al., arXiv 2016

# Example 2: in vivo PFC data




Zhou et.al., arXiv 2016

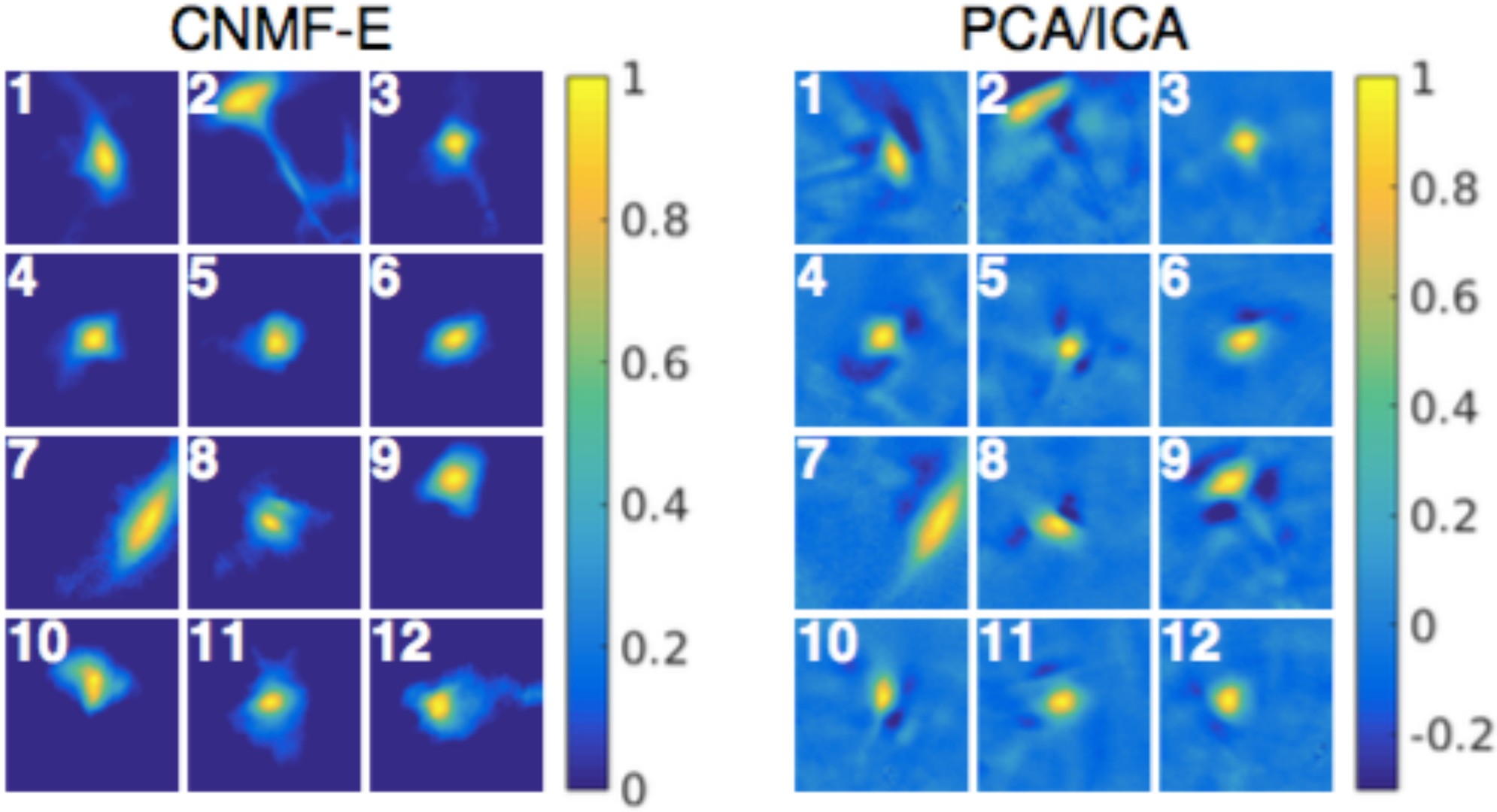
# Example 2: in vivo PFC data




The temporal traces extracted by CNMF-E have higher signal-to-noise ratios.

Zhou et.al., arXiv 2016

# Example 3: in vivo BNST data




Zhou et.al., arXiv 2016

1. BNST: bed nucleus of the stria terminals

# Example 3: in vivo BNST data

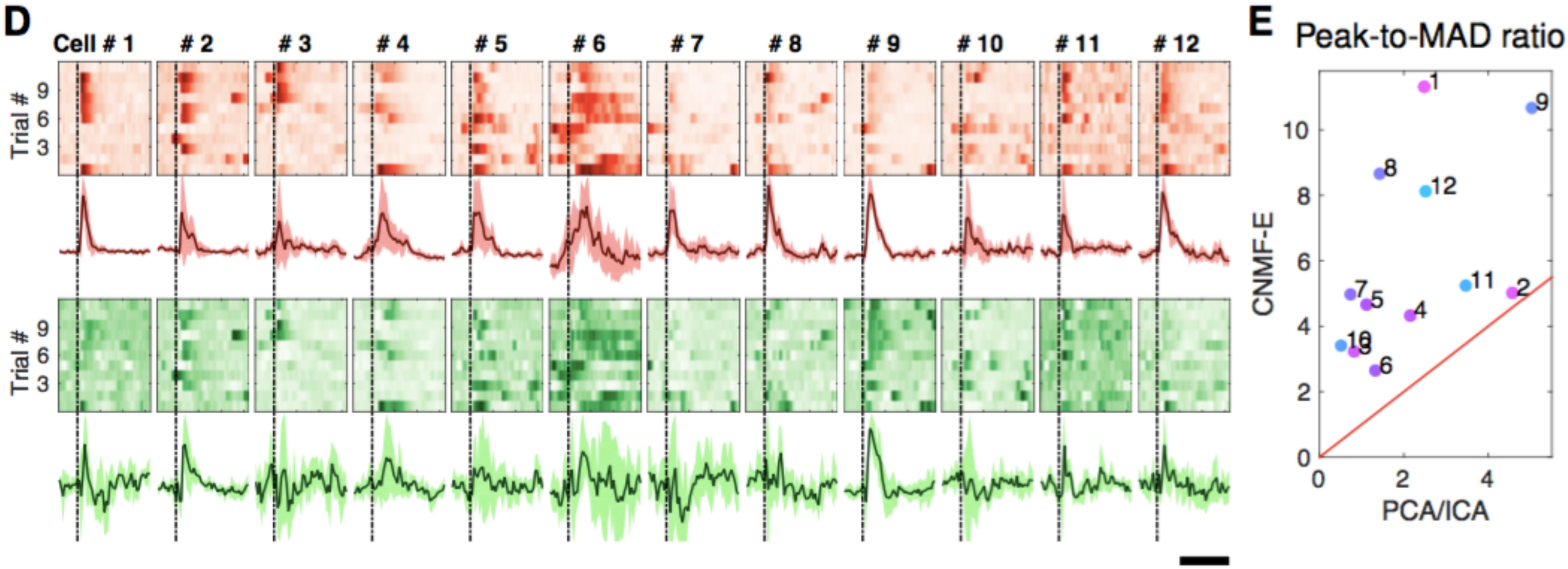



Dashed lines indicate electrical footshocks.

Zhou et.al., arXiv 2016

# Example 3: in vivo BNST data

Downstream analyses can significantly benefit from the improvements in the accuracy of source extraction




Zhou et.al., arXiv 2016

1. MAD: median absolute deviation, defined as the median of the absolute deviations from the data's median

# Summary

- Analyzing microendoscopic video data using CNMF-E
  - improve the accuracy of cell extraction
  - downstream analysis of calcium imaging data can significantly benefit from these improvements.

# Summary

- General calcium imaging data analysis
  - ROI analysis is simple and straightforward, but has limited performances.
  - CNMF is a general framework for simultaneously denoising, deconvolving and demixing calcium imaging data. Other matrix factorization methods are available as well (e.g., PCA/ICA, NMF, SHMF, SSTD).
  - Successful application of CNMF requires sufficient understanding of your data and modifying the model to fit your needs.

# Part II: Exercises

# Install CNMF-E

https://github.com/zhoupc/CNMF_E

# open demo script

```
>> cnmfe_setup
>> cd demos/
>> edit demo_endoscope.m
>>
```

demo_endoscope.m

```
%% clear workspace
clear; clc; close all;
global  d1 d2 numFrame ssub tsub sframe num2read Fs neuron neuron_ds ...
    neuron_full Ybg_weights; %#ok<NUSED> % global variables, don't change them manually

%% select data and map it to the RAM
% nam = '~/Dropbox/github/CNMF_E/demos/data_endoscope.tif';
cnmfe_choose_data;

%% create Source2D class object for storing results and parameters
Fs = 6;             % frame rate
ssub = 1;           % spatial downsampling factor
tsub = 1;           % temporal downsampling factor
gSig = 3;           % width of the gaussian kernel, which can approximates the average n
gSiz = 10;          % maximum diameter of neurons in the image plane. larger values are
neuron_full = Sources2D('d1',d1,'d2',d2, ... % dimensions of datasets
    'ssub', ssub, 'tsub', tsub, ...  % downsampleing
    'gSig', gSig,...    % sigma of the 2D gaussian that approximates cell bodies
    'gSiz', gSiz);      % average neuron size (diameter)
neuron_full.Fs = Fs;         % frame rate

% thresholds for merging neurons
```

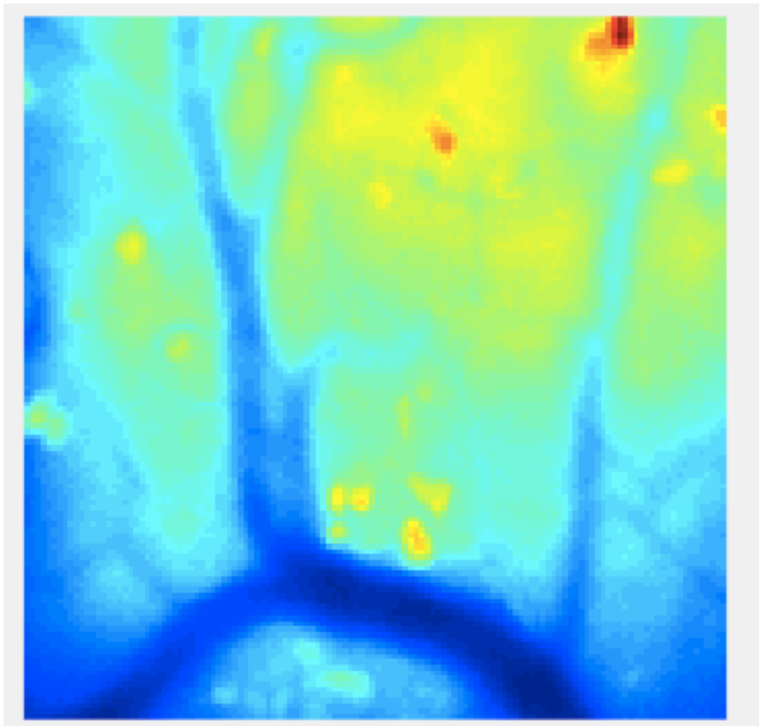
# select data

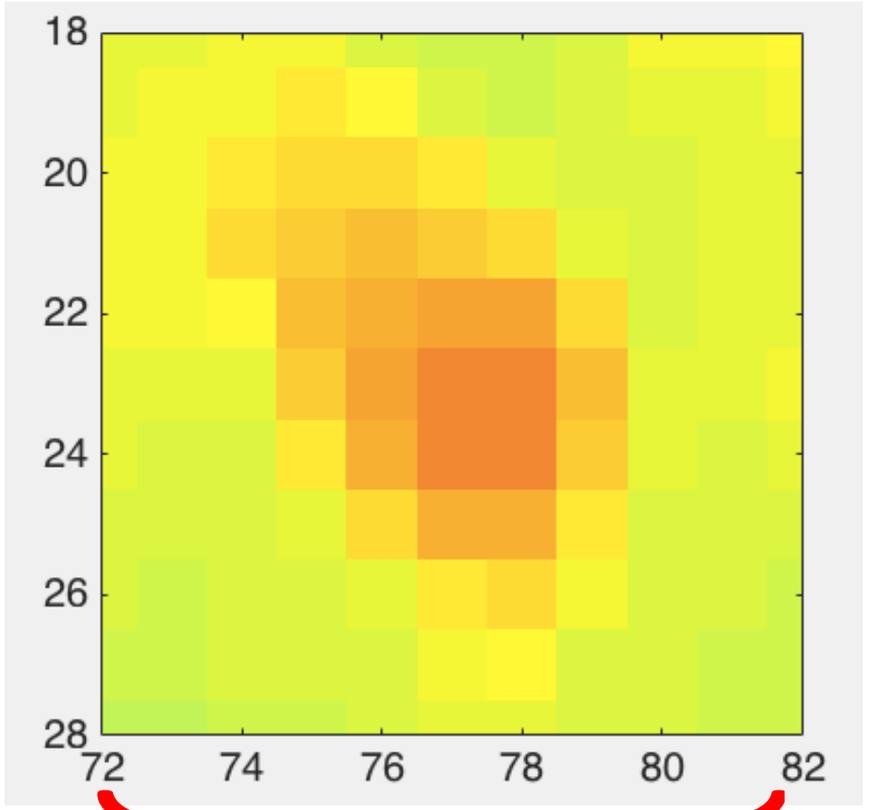
```
6     %% select data and map it to the RAM
7     % nam = '~/Dropbox/github/CNMF_E/demos/data_endoscope.tif';
8 -   cnmfe_choose_data;
9
```

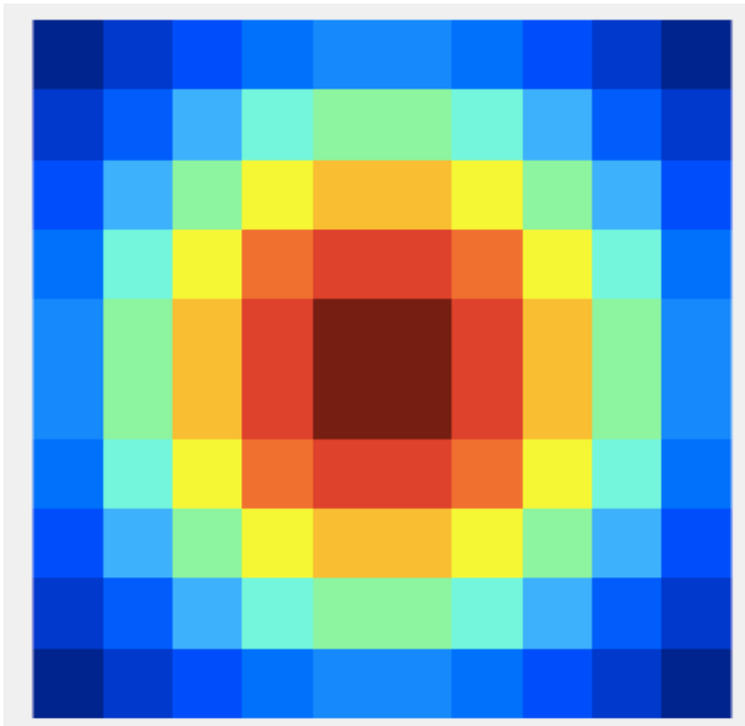
1. you can set the variable ***nam*** as the full path of your tif data.
2. If you didn't create a variable **nam** for storing file path, CNMF-E will ask you to select a file interactively.

When you load a tif data for the first time, CNMF-E will save the data into *.mat file. Then CNMF-E will always load *.mat data, instead of tif data.

# selection of gSig and gSiz





gSiz

gSiz/4 <= gSig <= gSiz/3

# downsampling data for fast initialization

```
9
10      %% create Source2D class object for storing results and parameters
11 -    Fs = 6;             % frame rate
12 -    ssub = 1;           % spatial downsampling factor
13 -    tsub = 1;           % temporal downsampling factor
14 -    gSig = 3;           % width of the gaussian kernel, which can approximates the average n
15 -    gSiz = 10;          % maximum diameter of neurons in the image plane. larger values are
16 -    neuron_full = Sources2D('d1',d1,'d2',d2, ... % dimensions of datasets
17          'ssub', ssub, 'tsub', tsub, ...  % downsampleing
18          'gSig', gSig,...    % sigma of the 2D gaussian that approximates cell bodies
19          'gSiz', gSiz);      % average neuron size (diameter)
20 -    neuron_full.Fs = Fs;          % frame rate
21
```

Run CNMF-E on down sampled data, then up-sample the results and use them as starting point of running CNMF-E on full resolution data.

# merging thresholds

```
21
22      % thresholds for merging neurons
23 -    merge_thr = [0.0001, 0.7, 0];     % thresholds for merging neurons
24      % corresponding to {sptial overlaps, temporal correlation of C, temporal correlation of
25 -    dmin = 1;  % minimum distances between two neuron centers. unit: pixel
26
```

```
178 -       cnmfe_quick_merge;                % run neuron merges
179 -       cnmfe_merge_neighbors;
```

1. merge neurons with high temporal correlation (merge_thr, cnmfe_quick_merge).
2. merge neurons that are too close (dmin, cnmfe_merge_neighbors).

I usually use the second method in the last iteration.

# options for running temporal deconvolution

```
38
39      %% options for running deconvolution
40 -    neuron_full.options.deconv_flag = true;  % deconvolve temporal traces
41 -    neuron_full.options.deconv_options = struct('type', 'ar1', ... % model of the calcium tr
42          'method', 'thresholded', ... % method for running deconvolution {'foopsi', 'constrai
43          'optimize_pars', true, ...  % optimize AR coefficients
44          'optimize_b', false, ... % optimize the baseline
45          'optimize_smin', true);  % optimize the threshold
46
```

When the rising time of one calcium transient is less than 1 frame, use ‘ar1’ type; otherwise, use ‘ar2’;

See Friedrich et al., Fast online deconvolution of calcium imaging data, PLOS Comp. Bio. (2017) for details of deconvolution algorithm.

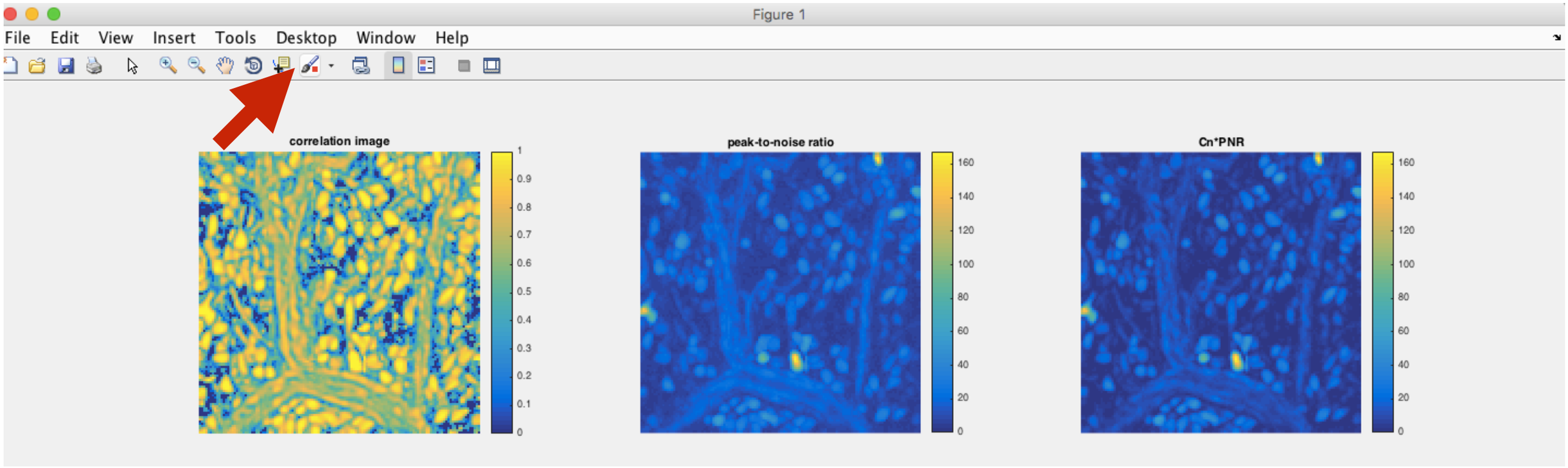
# load data

```
46
47      %% downsample data for fast and better initialization
48 -    sframe=1;                           % user input: first frame to read (optional, default:1)
49 -    num2read= numFrame;                 % user input: how many frames to read   (optional, defau
50
51 -    tic;
52 -    cnmfe_load_data;
53 -    fprintf('Time cost in downsapling data:     %.2f seconds\n', toc);
54
55 -    Y = neuron.reshape(Y, 1);           % convert a 3D video into a 2D matrix
56
```

sframe: index of the first frame
num2read: number of frames to be loaded.

# compute correlation image and PNR image

```
56
57      %% compute correlation image and peak-to-noise ratio image.
58 -    cnmfe_show_corr_pnr;     % this step is not necessary, but it can give you some...
59                               % hints on parameter selection, e.g., min_corr & min_pnr
60
```




pick some weak neurons and then use data curves to see their correlation values and PNR values.

# Initialization

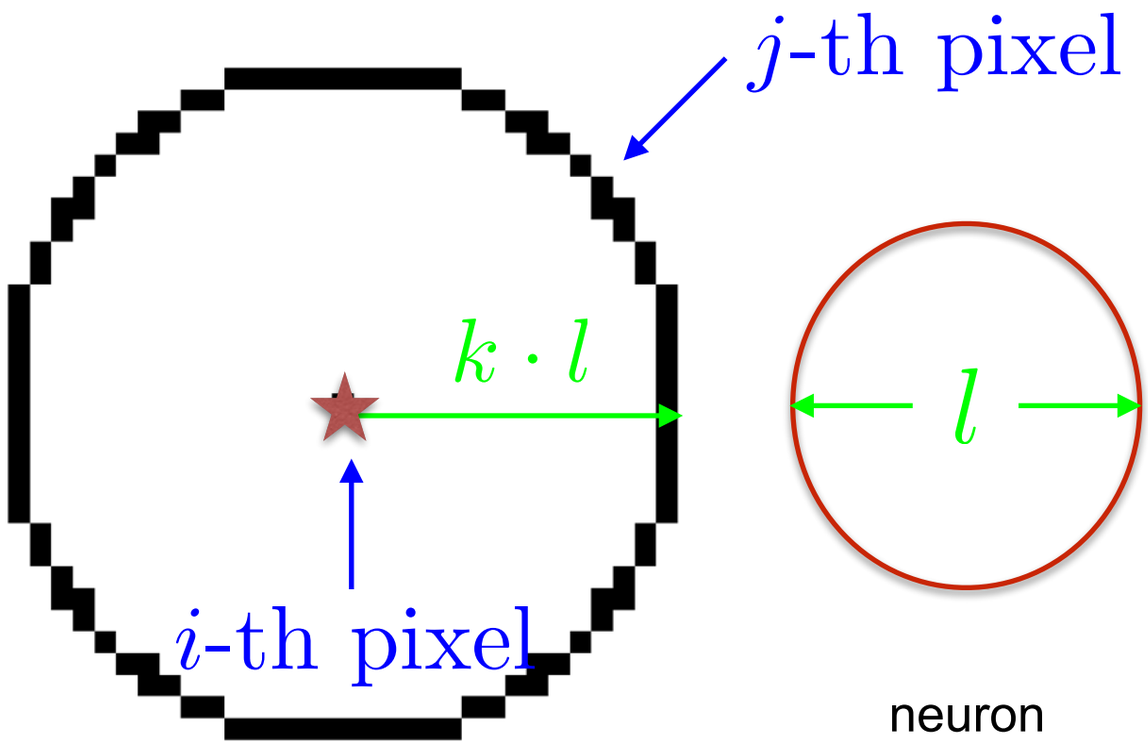
```
55
56      %% initialization of A, C
57      % parameters
58 -    debug_on = false;    % visualize the initialization procedue.
59 -    save_avi = false;    %save the initialization procedure as an avi movie.
60 -    patch_par = [1,1]*1; %1;  % divide the optical field into m X n patches and do initializ
61 -    K = []; % maximum number of neurons to search within each patch. you can use [] to searc
62
63 -    min_corr = 0.85;     % minimum local correlation for a seeding pixel
64 -    min_pnr = 20;        % minimum peak-to-noise ratio for a seeding pixel
65 -    min_pixel = 5;       % minimum number of nonzero pixels for each neuron
66 -    bd = 1;              % number of rows/columns to be ignored in the boundary (mainly for m
67 -    neuron.updateParams('min_corr', min_corr, 'min_pnr', min_pnr, ...
68          'min_pixel', min_pixel, 'bd', bd);
69 -    neuron.options.nk = 1;  % number of knots for detrending
70
71      % greedy method for initialization
72 -    tic;
73 -    [center, Cn, pnr] = neuron.initComponents_endoscope(Y, K, patch_par, debug_on, save_avi)
74 -    fprintf('Time cost in initializing neurons:     %.2f seconds\n', toc);
75
```

patch_par: divide the FOV into multiple patches when FOV is too large.

# parameters for estimating background

```
 97      % parameters, estimate the background
 98 -    spatial_ds_factor = 2;      % spatial downsampling factor. it's for faster estimation
 99 -    thresh = 10;     % threshold for detecting frames with large cellular activity. (mean of
100
101 -    bg_neuron_ratio = 1.5;  % spatial range / diameter of neurons
```

spatial_ds_factor: estimating B using fewer pixels.
bg_neuron_ratio:  k, the radius of the ring / gSiz



# three subproblems

## update B given A & C

```
117     %% udpate background
118     % estimate the background
119 -   tic;
120 -   cnmfe_update_BG;
121 -   fprintf('Time cost in estimating the background:        %.2f seconds\n', toc);
122     % neuron.playMovie(Ysignal); % play the video data after subtracting the background
123
```

## update C given A & B

```
128 -       neuron.updateTemporal_endoscope(Ysignal);
```

## update A given B & C

```
131 -       neuron.updateSpatial_endoscope(Ysignal, Nspatial, update_spatial_method);
132 -       neuron.trimSpatial(0.01, 3); % for each neuron, apply imopen first and then remo
```

# Interventions

```
178 -       cnmfe_quick_merge;                  % run neuron merges
179 -       cnmfe_merge_neighbors;
```

merge neurons.

```
168
169 -      neuron.viewNeurons([], neuron.C_raw);
```

display each neurons and allow you delete/split/trim neurons.

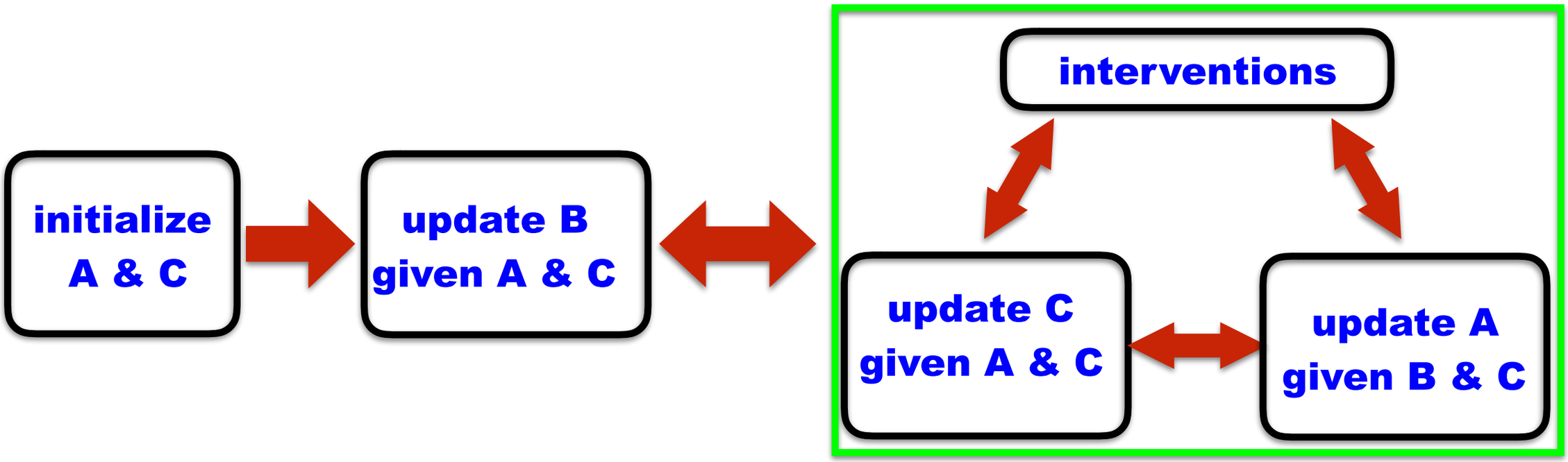
```
138
139        %% pick neurons from the residual (cell 4).
140 -      if miter==1
141 -          neuron.options.seed_method = 'auto'; % methods for selecting seed pixels {'auto'
142 -          [center_new, Cn_res, pnr_res] = neuron.pickNeurons(Ysignal - neuron.A*neuron.C,
143 -      end
144
```

pick neurons from the residuals.

# Interventions

```
154
155       %% apply results to the full resolution
156 -     if or(ssub>1, tsub>1)
157 -         neuron_ds = neuron.copy();  % save the result
158 -         neuron = neuron_full.copy();
159 -         cnmfe_full;
160 -         neuron_full = neuron.copy();
161 -     end
162
```

run CNMF-E on data with full resolution.


initialize
A & C
update B
given A & C
interventions
update C
given A & C
update A
given B & C

# Future plans

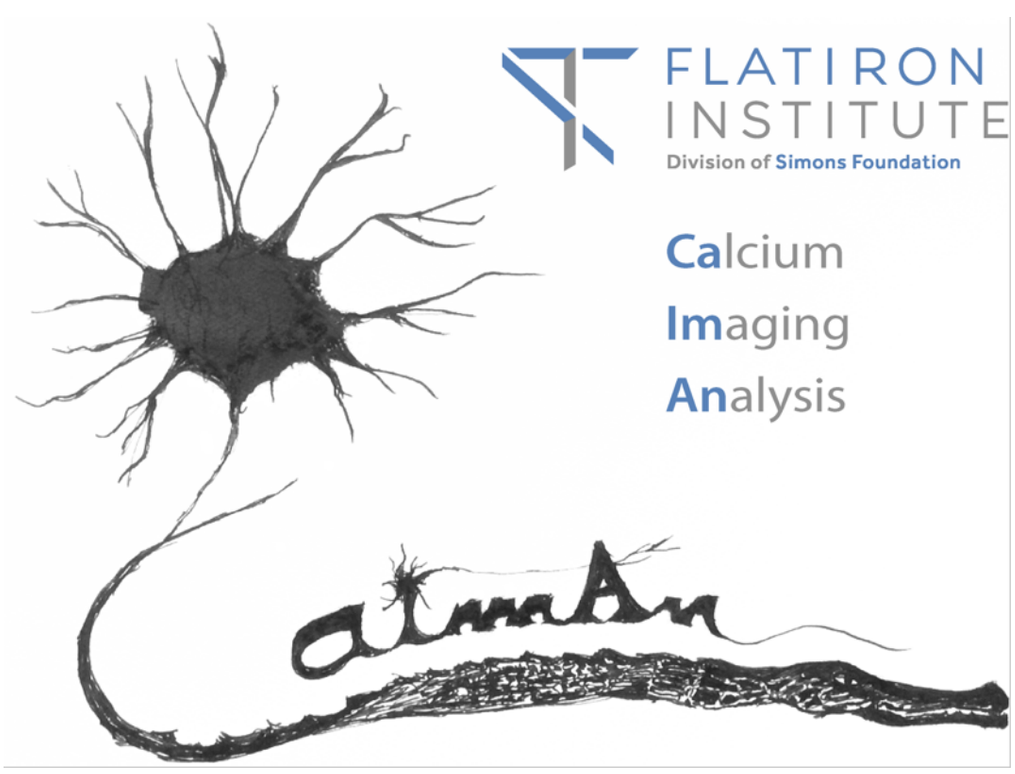
- A complete suite for processing calcium imaging data, including motion correction as well.
- Support running CNMF-E on clusters for fast parallel computing
- Optimize algorithms for running CNMF-E in batch model and remove the bottleneck of memory usage.
- Allow running CNMF-E in real-time for closed-loop experiments.



# Future plans

For python users:

Integrate CNMF-E into an open-source package CaImAn (**Ca**lcium **Im**aging **An**alysis )




https://github.com/simonsfoundation/CaImAn

# Acknowledgements






@Columbia



@CMU



@UCSF

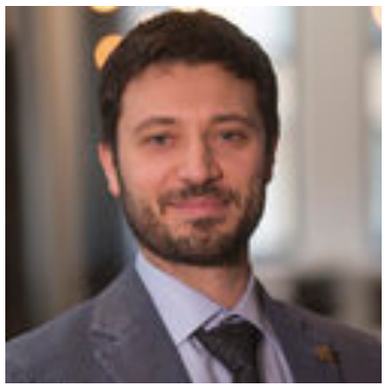




@UNC




@Harvard

@Simons Foundation

# Acknowledgements

## All CNMF-E users

https://beat-ica.slack.com/

