Fast spike train inference from calcium imaging

Joshua T. Vogelstein, others, Liam Paninski

June 19, 2009

Contents

Abstract

1 Introduction

Simultaneously imaging large populations of neurons using calcium sensors is becoming increasingly popular [1, 2, 3, 4], especially as the signal-to-noise-ratio (SNR) of genetic sensors continues to improve [5, 6, 7]. While this technology facilitates acquiring data from an unprecedented number of neurons simultaneously, it is not yet a panacea: there is a trade-off between the quantity of neurons, and the quality of data. While one can now acquire data from ~ 100 neurons within a single experimental session, the data for each neuron is relatively poor. In comparison with extracellular electrophysiology, the data from calcium imaging has reduced (i) SNR and (ii) temporal resolution. Thus, relatively few studies have been able to use this approach to date to ask *quantitative* questions about the relationship between spike trains, and, for example, sensory stimuli [8, 9, 10, 11, 12, 13, 14].

A number of groups have proposed spike inference algorithms to facilitate using calcium imaging to ask quantitative questions about spike timing. Finding the most likely spike train, given a calcium movie, is computationally intractable because we have to perform a search over all possible spike trains, and the number of possible spike trains scales exponentially with the number of image frames, T (more specifically, assuming only a single spike can occur per frame, we have 2^T possible spike trains). To circumvent this problem, various groups have developed distinct strategies. For instance, Greenberg et al. [15] reduced the search space by establishing heuristics to preprocess the data, and effectively exclude many image frames from the search space. In our previous work [16], we developed a sequential Monte Carlo method to efficiently sample spike trains given the fluorescence data, which is guaranteed to perform optimally given our model, which incorporates saturation, refractoriness, and stimulus dependent effects (this approach has the added advantage of providing a measure of uncertainty as well). Unfortunately, both these approaches are relatively slow, as they still require searches over large spaces. Holekamp et al. [17] took a very different strategy, by performing the optimal linear deconvolution (i.e., the Wiener filter) on the fluorescence data. This approach may be thought of as finding the *maximum a posteriori* (MAP) spike train.

The present work also develops a deconvolution filter, but has several advantages over the Wiener filter. First, we impose a non-negative constraint on the solution. Because we know that spikes are non-negative events, this is a desirable constraint to impose on our inference procedure. In practice, this constraint regularizes the resulting inference [18, 19, 20], by combating against "ringing" overfitting effects. Second, the computational time of our filter scales linearly with T, whereas the Wiener filter typically scales according to $T \log T$. Thus, we have named our approach the FAst Non-negative Deconvolution (FAND) filter. This FAND filter could be applicable in a number of scientific investigations from myriad fields, and is closely related to the problem of non-negative matrix factorization, a problem currently receiving much attention from the machine learning community [21, 18, 19, 22, 23].

The remainder of this paper is organized as follows. In Section 2, we describe our parametric model, and the FAND filter. Section 3 first provides our main result: we can utilize FAND to filter calcium fluorescence data, with better results than the optimal linear filter. We then generalize the FAND filter in a number of directions. First, instead of assuming the data is a 1-dimensional fluorescence time series, we operate on all the pixels within a region-of-interest (ROI), which can significantly improve the SNR of our inference. Second, by embedding our FAND filter into a pseudo-expectation-maximization algorithm, we can learn the parameters of our model without any training data (i.e., without requiring simultaneous imaging and electrophysiology). To demonstrate the utility of this approach, we compare our filter and the Wiener filter's output on in vitro data. Third, when imaging a large population of neurons simultaneously, sometimes segmenting the image to obtain one neuron per ROI is difficult. Therefore, we show how approach can be further generalized to deal with such a scenario. We then apply our filter to a movie of ~ 50 neurons recorded simultaneously in vitro, to indicate that this approach works "out-of-the-box" on segmented data. Fourth, we modify our model to handle data collected in vivo using genetic sensors. Specifically, this means allowing for a slow rise time and poisson observations. Finally, in the discussion, we show how to incorporate non-linear saturation into the FAND filter, and discuss how the output of our FAND filter can be used to initialize our more powerful (but slower) algorithms.

2 Methods

2.1 Model

We start by assuming a very simple generative model, relating the hidden neural activity (spike trains and intracellular calcium concnetrations) and the observations (fluorescence movies). More specifically, we start by assuming we have collected a 1-dimensional fluorescence trace, $\boldsymbol{F} = (F_1, \ldots, F_T)$ from a neuron. At time t, the fluorescence measurement, F_t is a linear-Gaussian function of the intracellular calcium concentration at that time, C_t:

$$F_t = \alpha(C_t + \beta) + \sigma\varepsilon_t, \qquad \varepsilon_t \overset{iid}{\sim} \mathcal{N}(0,1) \tag{1}$$

The scale, α, absorbs all experimental variables impacting the scale of the signal, including number of sensors within the cell, photon/$\Delta[\text{Ca}^{2+}]$ per sensor, amplification of imaging system, etc. Similarly, the offset, β, absorbs baseline calcium concentration of the cell, background fluorescence of the fluorophore, imaging system offset, etc. The standard deviation, σ, results from calcium fluctuations independent of spiking activity, fluorescence fluctuations independent of calcium, and imaging noise. These three parameters therefore correspond to a number of simplifying assumptions, that we will relax in Section 3.

We further assume that the intracellular calcium concentration, C_t, jumps after each spike, and subsequently decays back down to rest with time constant, τ, yielding $\tau(C_t - C_{t-1})/\Delta = -C_{t-1} + n_t$. Rearranging (and rescaling) a bit, we have:

$$C_t = \gamma C_{t-1} + n_t, \qquad n_t \overset{iid}{\sim} \text{Poisson}(\lambda\Delta) \tag{2}$$

where n_t indicates the number of times the neuron spiked at time t, and $\gamma = 1 - \Delta/\tau$. [1] Note that C_t does not refer to absolute intracellular concentration of calcium but rather, a relative measure. The assumed linearity of our model precludes the possibility of determining calcium in absolute terms (but see Section 4.1 for a modified model). Figure 1 depicts a schematic illustration of this model. Note that the model, Eqs. (1) and (2) is a discrete-time state-space model. Importantly, this model differs from the standard models in that we have a non-Gaussian noise term, n_t. Thus, standard tools, such as the Kalman filter-smoother, cannot be applied directly. We therefore develop a novel algorithm to handle problems of this nature, as described below.

[1] This follows from writing (2) as $\tau \frac{C_t - C_{t-1}}{\Delta} = -C_{t-1} + n_t$.

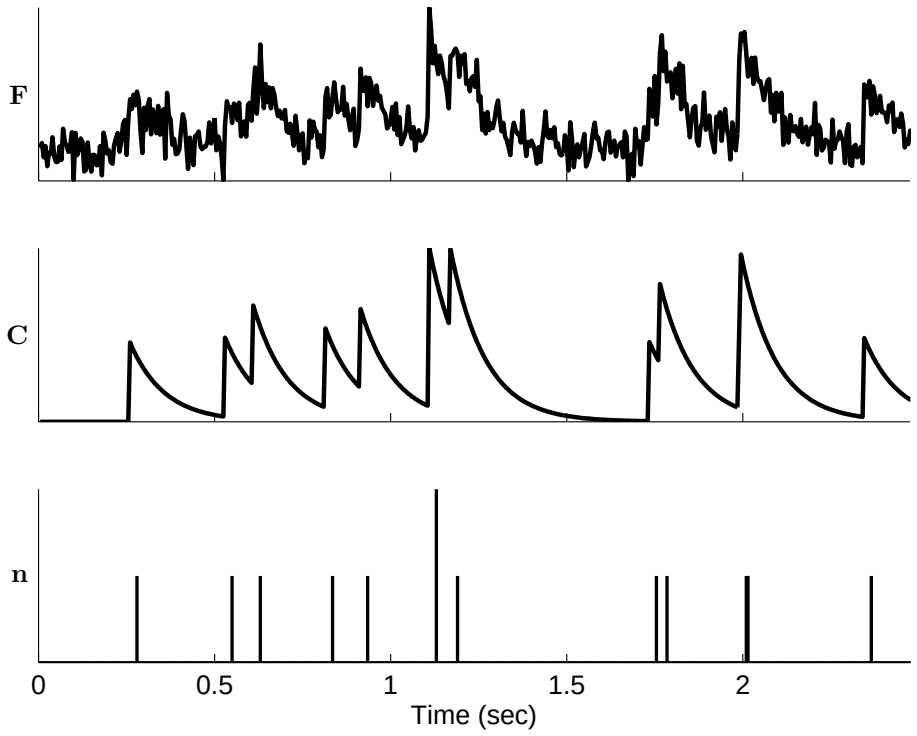

Figure 1: A schematic illustrating our model. The spike train (bottom panel) is convolved with an exponential with time constant τ to obtain the calcium dynamics (middle panel). The fluorescence observations are simply the calcium dynamics, plus zero mean Gaussian noise (with variance σ^2). Parameters: $\Delta = 5$ msec, $\alpha = 1$ photons/μM, $\beta = 0$ unitless, $\sigma = 0.25$ photons, $\tau = 100$ msec, $\lambda = 5$ Hz.

2.2 Goal

Given the above model, our goal is to find the maximum *a posteriori* (MAP) spike train, i.e., the most likely spike train, $\boldsymbol{n}$, given the fluorescence measurements, $\boldsymbol{F}$. Formally, we have:

$$\boldsymbol{n}_{MAP} = \underset{n_t \in \mathbb{N}_0 \forall t}{\operatorname{argmax}} P[\boldsymbol{n}|\boldsymbol{F}] \tag{3a}$$

$$= \underset{n_t \in \mathbb{N}_0 \forall t}{\operatorname{argmax}} P[\boldsymbol{F}|\boldsymbol{n}]P[\boldsymbol{n}] \tag{3b}$$

$$= \underset{n_t \in \mathbb{N}_0 \forall t}{\operatorname{argmax}} \prod_{t=1}^{T} P[F_t|C_t]P[n_t] = \underset{n_t \in \mathbb{N}_0 \forall t}{\operatorname{argmax}} \sum_{t=1}^{T} \big(\log P[F_t|C_t] + \log P[n_t] \big) \tag{3c}$$

where 3a is the definition of the MAP spike train, 3b follows from Bayes' Rule, 3c follows from the state-space nature of our problem, $\mathbb{N}_0$ is the set of natural numbers (i.e., $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$), T is the number of time steps in the recording, and C_t is implicitly a function of n_t. Plugging Eqs. (1) and (2) into Eq. (3c) yields:

$$\boldsymbol{n}_{MAP} = \underset{n_t \in \mathbb{N}_0 \forall t}{\operatorname{argmin}} \sum_{t=1}^{T} \left(\frac{1}{2\sigma^2} (F_t - \alpha(C_t + \beta))^2 - n_t \log \lambda\Delta + \log n_t! \right), \tag{4}$$

Unfortunately, solving Eq. (4) exactly is computationally intractable, as it requires a nonlinear search over all possible spike trains (and since $n_t \in \mathbb{N}_0$, there are an infinite number of them). We could restrict our search space by imposing

an upper bound, k, on the number of spikes within a frame. However, in that case, the computational complexity scales *exponentially* with the number of image frames — more specifically, requires $O(k^T)$ time — which for pragmatic reasons is intractable. Thus, we approximate Eq. (4), by modifying Eq. (2), replacing the Poisson distribution with one that yields a log-concave problem. This reduces the algorithmic complexity from requiring a search over $O(k^T)$ possible spike trains, to some kind of gradient ascent solution.

Two distributions naturally arise as possible approximations to a Poisson: (i) exponential, and (ii) Gaussian. As depicted in Figure 2 an exponential distribution (dashed gray line) is an excellent approximation to the Poisson distribution (solid black line), when λ is small (left panel). On the other hand, when λ is large, a Gaussian distribution (dash-dotted gray line) closely approximates a Poisson when λ is large (right panel). Thus, naïvely, it seems as though it may be desirable to use a Gaussian approximation when the neuron is firing with a high firing rate, and approximate the Poisson with an exponential when the neuron is firing sparsely. The Wiener filter is, in fact, the optimal filter given the Gaussian approximation [24] (see [17] for an application of the Wiener filter to this problem). Below, we develop an algorithm to perform inference given the exponential distribution.

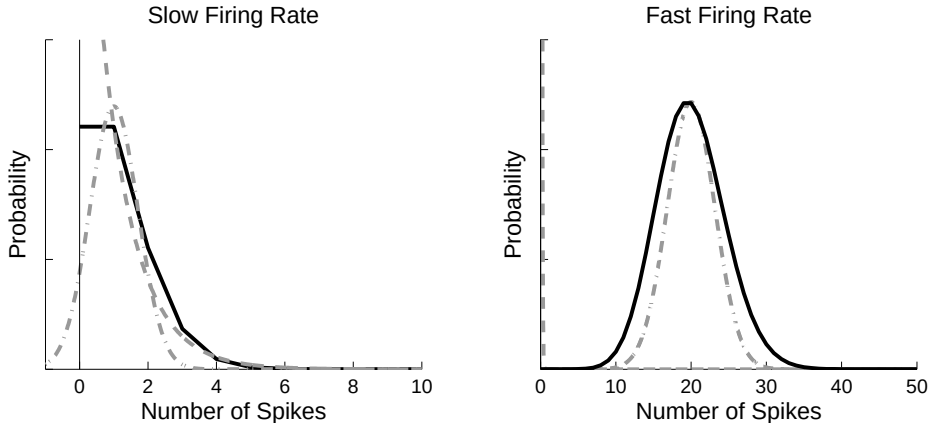

Figure 2: Approximating a Poisson distribution. Left panel: when λ is small (e.g., ≈ 1), an exponential distribution (dashed gray line) approximates the Poisson distribution (solid black line) well, but a Gaussian distribution (dash-dotted gray line) does not. Right panel: when λ is large (e.g., ≈ 20), the inverse is true.

2.3 Inference

FAst Non-negative Deconvolution (FAND) Filter Our goal here is to develop an algorithm to efficiently approximate $\widehat{\boldsymbol{n}}_{MAP}$. The main contribution of this work demonstrating that we can substitute the Poisson distribution with an exponential distribution, reducing computational complexity from $O(k^T)$ to $O(T)$. Formally, we replace Eq. (2) with:

$$C = \gamma C_{t-1} + n_t, \qquad n_t \overset{iid}{\sim} \text{Exponential}(\lambda\Delta) \tag{5}$$

where the only difference between Eq. (2) and Eq. (5) is the distribution on n_t. Plugging this change into Eq. (**??**) yields

$$\boldsymbol{n}^{FAND} = \underset{n_t>0\,\forall t}{\text{argmax}} \sum_{t=1}^{T} \left(\frac{1}{2\sigma^2}(F_t - \alpha(C_t + \beta))^2 + n_t\lambda\Delta \right) \tag{6}$$

where the constraint on $\boldsymbol{n}$ has been relaxed from $n_t \in \mathbb{N}_0$ to $n_t \geq 0$ (since the support of an exponential distribution is the non-negative number line). Note that this is a common approximation technique in the machine learning literature [25], as it is the closest convex relaxation to its non-convex counterpart. While this convex relaxation makes the problem tractable, the "sharp" threshold imposed by the non-negativity constraint prohibits the use of standard gradient ascent techniques [26]. We therefore take an "interior-point" (or "barrier") approach, in which we drop the sharp

threshold, and add a barrier term, which must approach $-\infty$ as n_t approaches zero (e.g., $-\log n_t$) [26]. By iteratively reducing the weight of the barrier term, we are guaranteed to converge to the correct solution [26]. Thus, our goal is to efficiently solve:

$$\widehat{\boldsymbol{n}}_z = \underset{n_t \forall t}{\operatorname{argmin}} \sum_{t=1}^{T} \left(\frac{1}{2\sigma^2}(F_t - \alpha(C_t + \beta))^2 + n_t \lambda \Delta - z \log(n_t) \right), \tag{7}$$

Since spikes and calcium are related to one another via a simple linear transformation, namely, $n_t = C_t - \gamma C_{t-1}$, we may rewrite Eq. (7) in terms of $\boldsymbol{C}$:

$$\widehat{\boldsymbol{C}}_z = \underset{C_t - \gamma C_{t-1} \geq 0 \forall t}{\operatorname{argmin}} \sum_{t=1}^{T} \left(\frac{1}{2\sigma^2}(F_t - \alpha(C_t + \beta))^2 + (C_t - \gamma C_{t-1}) \lambda \Delta - z \log(C_t - \gamma C_{-1}) \right). \tag{8}$$

The concavity of Eq. (8) facilitates utilizing any number of techniques guaranteed to find the global optimum. The fact that the argument of Eq. (8) is twice differentiable, suggests that we use the Newton-Raphson technique. Importantly, the state-space nature of this problem yields a particularly efficient approach. Specifically, note that the Hessian is *tridiagonal*, which is clear upon rewriting (8) in matrix notation:

$$\widehat{\boldsymbol{C}}_z = \underset{\boldsymbol{MC} \geq \boldsymbol{0}}{\operatorname{argmin}} \frac{1}{2\sigma^2} \left\| \boldsymbol{F} - \alpha(\boldsymbol{C} + \beta) \right\|^2 + (\boldsymbol{MC})^{\mathsf{T}} \boldsymbol{\lambda} - z \log(\boldsymbol{MC})^{\mathsf{T}} \boldsymbol{1}, \tag{9}$$

where $\boldsymbol{M} \in \mathbb{R}^{T \times T}$ is a bidiagonal matrix, $\boldsymbol{MC} \geq \boldsymbol{0}$ indicates that every element of $\boldsymbol{MC}$ is greater than or equal to zero, $^{\mathsf{T}}$ indicates transpose, $\boldsymbol{1}$ is a T dimensional column vector, $\boldsymbol{\lambda} = \lambda \Delta \boldsymbol{1}^{\mathsf{T}}$, and $\log(\cdot)$ indicates an element-wise logarithm. Note that Eq. (9) follows from writing $\boldsymbol{n}$ in terms of $\boldsymbol{M}$ and $\boldsymbol{C}$:

$$\boldsymbol{MC} = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots \\ 1 & -\gamma & 0 & \cdots & \cdots \\ 0 & 1 & -\gamma & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & -\gamma \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ \vdots \\ C_T \end{bmatrix} = \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ \vdots \\ n_T \end{bmatrix} = \boldsymbol{n} \tag{10}$$

Thus, a little bit of calculus yields our update algorithm for $\boldsymbol{C}_z$:

$$\widehat{\boldsymbol{C}}_z \leftarrow \widehat{\boldsymbol{C}}_z + s\boldsymbol{d} \tag{11a}$$

$$\boldsymbol{H}\boldsymbol{d} = \boldsymbol{g} \tag{11b}$$

$$\boldsymbol{g} = -\frac{\alpha}{\sigma^2}(\boldsymbol{F} - \alpha(\widehat{\boldsymbol{C}}_z^{\mathsf{T}} + \beta)) + \boldsymbol{M}^{\mathsf{T}} \boldsymbol{\lambda} - z \boldsymbol{M}^{\mathsf{T}} (\boldsymbol{M} \widehat{\boldsymbol{C}}_z)^{-1} \tag{11c}$$

$$\boldsymbol{H} = \frac{\alpha^2}{\sigma^2} \boldsymbol{I} + z \boldsymbol{M}^{\mathsf{T}} (\boldsymbol{M} \widehat{\boldsymbol{C}}_z)^{-2} \boldsymbol{M} \tag{11d}$$

where s is the step size, $\boldsymbol{d}$ is the step direction, and $\boldsymbol{g}$ and $\boldsymbol{H}$ are the gradient (first derivative) and Hessian (second derivative) of the argument in Eq. (9) with respect to $\boldsymbol{C}$, respectively, and the exponents indicate element-wise operations. Note that we use "backtracking linesearches", meaning that for each iteration, we find the maximal s that is (i) between 0 and 1 and (ii) decreases the likelihood.

Typically, implementing Newton-Raphson requires inverting the Hessian, i.e., $\boldsymbol{d} = \boldsymbol{H}^{-1}\boldsymbol{g}$, a computation consuming $O(T^3)$ time. Already, this would be a drastic improvement over the most efficient algorithm assuming Poisson spikes, which require $O(k^T)$ time (where k is the maximum number of spikes per frame). Here, because $\boldsymbol{M}$ is bidiagonal, the Hessian is tridiagonal, the solution may be found in $O(T)$ time via standard banded Gaussian elimination techniques (which can be implemented efficiently in Matlab using $\boldsymbol{H} \backslash \boldsymbol{g}$). In other words, the above approximation and inference algorithm reduces computations from exponential time to *linear* time. We refer to this fast algorithm for solving (6) the FAst Nonnegative Deconvolution (FAND) filter.

Fast Wiener Filter Instead of replacing the Poisson distribution on spikes with an exponential, we can replace it with a Gaussian:

$$C = \gamma C_{t-1} + n_t, \qquad n_t \overset{iid}{\sim} \mathcal{N}(\lambda\Delta, \lambda\Delta) \tag{12}$$

which, when plugged into Eq. (3c) yields

$$\boldsymbol{n}^{Wiener} = \underset{n_t}{\operatorname{argmax}} \sum_{t=1}^{T} \left(\frac{1}{2\sigma^2}(F_t - \alpha(C_t + \beta))^2 + \frac{1}{2\lambda\Delta}(n_t - \lambda\Delta)^2 \right) \tag{13}$$

Using the same tridiagonal trick as above, we can solve Eq. (13) using Newton-Raphson once (since we have a quadratic problem here, see Appendix A for details). Because we know only positive spikes are possible, at times we will also consider, $[\text{Wiener}]_+$, which is the Wiener filter half-wave rectified, i.e., all sub-zero values are set to zero.

2.4 Assessment

Let $\widehat{\boldsymbol{n}}$ be the inferred spike train, using some algorithm, and $\boldsymbol{n}$ be the true spike train. When spiking is sparse (i.e., at most, one spike per frame), a reasonable measure of inference quality is the effective signal-to-noise ratio, which we define as the squared size of $\widehat{\boldsymbol{n}}$ when there is a spike, divided by the squared size of $\widehat{\boldsymbol{n}}$ when there is no spike:

$$\text{eSNR}_{\widehat{\boldsymbol{n}}} = \frac{\sum_{t|n_t=1} \widehat{n}_t}{\sum_{t|n_t=0} \widehat{n}_t} \tag{14}$$

When the neuron is emits many spikes per frame, eSNR is not meaningful. Instead, we compute the mean squared error between the magnitude of inferred spikes and the true spikes:

$$\text{MSE}_{\widehat{\boldsymbol{n}}} = \frac{1}{T} \sum_t (n_t - \widehat{n}_t)^2 \tag{15}$$

3 Results

3.1 Main Result

The main result of this paper is that we can approximate $\boldsymbol{n}^{MAP}$ in *linear* time, whereas an exact solution would require exponential time. Fig. 3 shows two examples of running the FAND filter on simulated data. On the left, the data is simulated according to Eqs. (1) and (2), with a low expected firing rate (5 Hz). The FAND filter performs very well (middle left panel): when spikes occured (gray downward facing triangles), the FAND filter's output is relatively high, and in the absence of a spike, the FAND filter's output is relatively low. The height of the "spikes" in $\boldsymbol{n}^{FAND}$ can be thought of as the probability of a spike having occurred. The Wiener filter does not perform as well on this data. Specifically, $\boldsymbol{n}^{Wiener}$ exhibits a "ringing" effect, in which the inference oscillates around zero in the absence of true spikes. This occurs because negative spikes improve the inference, if negative spikes are allowed. By constraining our inference to be non-negative (for the FAND filter), we completely circumvent this problem.

It is unsurprising that the FAND filter significantly outperforms the Wiener filter when spiking is sparse, given that the exponential is a much better approximation to the Poisson in this regime (c.f. Figure 2). However, even in the fast firing rate scenario, where the Gaussian approximation is far more accurate, the FAND filter performs relatively well, as depicted in the right panels of Figure 3. Note however that the computational time for computing the FAND filter scales linearly with T, i.e. is $O(T)$, whereas the naïve implementation of the Wiener filter requires $O(T \log T)$ (but see Appendix **??** for an implementation of the Wiener for that only requires $O(T)$).

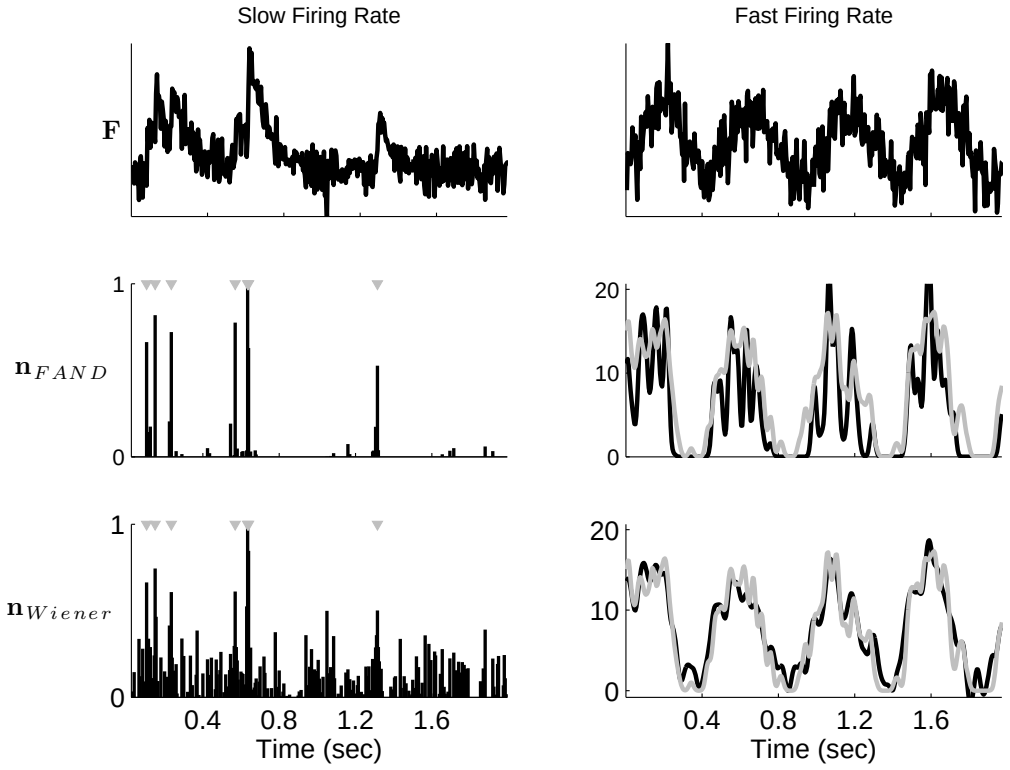

Figure 3: A simulation demonstrating the performance of the FAND filter in different firing regimes. The left panels show that in the sparse firing regime, the FAND filter outperforms the Wiener filter in terms of SNR. This follows because an exponential is a closer approximation to a Poisson than a Gaussian, when spiking is sparse. The right panels show that both approximations are good in the fast firing regime. Top left panel: fluorescence time series for a neuron with a slow firing rate. Middle left panel: the FAND filter's inferred spike train. Bottom left panel: Wiener filter's inferred spike train. Note that (i) the Wiener filter does not impose a non-negativity constraint, and (ii) the effective SNR of the Wiener filter in this example is worse than the FAND filter's. Top right panel: same as top left panel, for a neuron with a high firing rate. Middle right panel: the FAND filter's inferred spike train smoothed with a Gaussian kernel for visualization purposes (black line), and the true spike train smoothed with the same Gaussian kernel (gray line). Bottom right panel: same as middle right panel, but with the Wiener filter. Parameters for left panels: same as in Figure 1. Parameters for right panels: same as Figure 1, except: $\sigma = 8$ photons, $\lambda = 500$ Hz.

To quantify the relative quality of these inference algorithms in the sparsely spiking regime, we compute the effective signal-to-noise ratio (eSNR), defined as the average squared magnitude of the inferred spiked during frames in which there was a spike, divided by the same quantity computed in frames lacking a spike. The left panels of Figure 4 show how the eSNR varies as a function of the magnitude of the noise on observations (top left panel), and the expected firing rate (bottom left panel). The FAND filter's inference (blue line) dominates the Wiener filter (red line), as well as the post-hoc half wave rectifier Wiener filter, $[\text{Wiener}]_+$ (green line), and a simple dF/F (turquoise line).

When the neuron is exhibiting a high firing rate, we compute the mean-squared error (MSE) between the inferred number of spikes, and the true number of spikes. The top right panel of Figure 4 shows that when noise is relatively low, the FAND filter performs about as well as the Wiener filter. However, in the high noise regime, the Wiener filter clearly outperforms the FAND filter.

To verify that both our implementations of the FAND filter and the Wiener filter scale linear with respect to the number of image frames, the bottom right panel of Figure 4 shows a such a linear relationship (on a log-log scale).

The take home message from Figure 4 is that when we expect < 1 spike per frame, the FAND filter significantly outperforms the Wiener filter, regardless of variance of the observation noise. Furthermore, because of our linear time algorithm, filtering around 50,000 image frames requires only about 1 second on a standard Apple laptop. Below, we improve on our inference quality by generalizing our model in a number of ways.

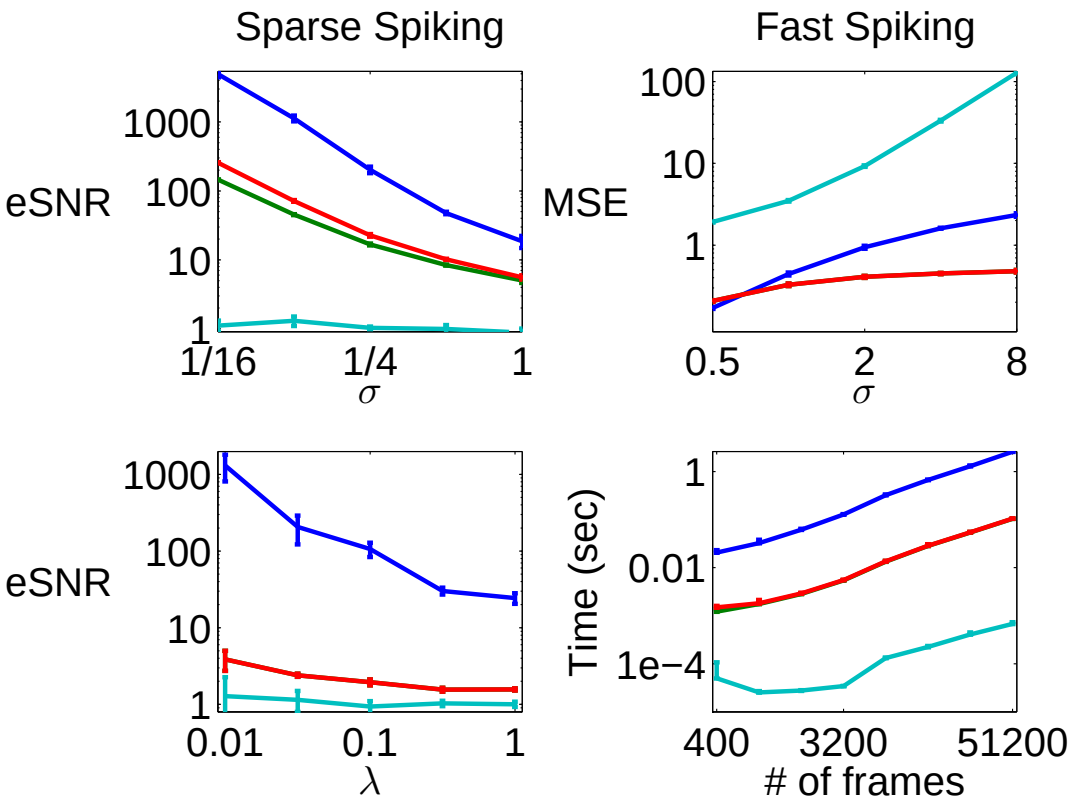

Figure 4: Quantitative assessment of the FAND filter's inference quality and speed. The left columns show that the FAND filter outperforms the Wiener filter in terms of eSNR — Eq. (14) — regardless of variance of observation noise (top) and expected firing rate (bottom). The top right panel show that when noise is low, even when firing rates are relatively (e.g. $\lambda\Delta = 20$ Hz), the FAND filter and Wiener filter both perform well. Both algorithms scale linearly with the number of image frames, meaning that it takes about 1 second of computational time per 50,000 image frames, using either algorithm. Simulation details: 5 simulations for each point on all the plots, mean (solid line) and standard deviation (bars) are shown for each. Parameters: $\Delta = 0.005$ sec, $\alpha = 1$, $\beta = 0$. Top left: $\lambda = 1$ Hz, $\tau = 1$ sec. Top right: $\lambda = 10$ Hz, $\tau = 1$ sec. Bottom left: $\sigma = 0.25$, $\tau = 0.5$ sec. Bottom right: $\sigma = 0.25$, $\tau = 0.1$ sec, $\lambda = 1$ Hz.

Finally, often one is interested in understanding the relationship between spike trains and the environment. Therefore, we simulated a neuron whose spiking activity was a function of a 5-dimensional external stimulus. More specifically, we let $\lambda_t = \boldsymbol{k}^\mathsf{T}\boldsymbol{x}_t$, where $\boldsymbol{k}$ is a 5-dimensional linear kernel, $\boldsymbol{x}_t$ is the stimulus at time t, and $P(n_t) =$Poisson$(\lambda_t\Delta)$. We then computed the maximum likelihood estimate of the linear kernel, $\widehat{\boldsymbol{k}} = \widehat{\boldsymbol{n}}\boldsymbol{x}^\mathsf{T}(\boldsymbol{x}\boldsymbol{x}^\mathsf{T})^{-1}$. Importantly, the simulation was constructed to incorporate both sparse spiking and fast spiking periods, much like sensory neurons have periods of quiescence, followed by stimulus driven bursts. Thus, neither the FAND filter nor the Wiener filter's assumptions are entirely appropriate. Figure 5 shows the results of this simulation. The true kernel, kernel estimated using the true spike times, and kernel estimated using the FAND filter, are nearly overlapping (black, gray, and blue lines, respectively). The kernel estimated using the Wiener filter output, however, is effectively flat (red line). Post-hoc half-wave rectification of the Wiener filter output does not improve its ability to estimate this kernel (green line — completely obscured by the unrectified Wiener filter output). Finally, dF/F does not yield anything useful at all (turquoise line). This simulation provides further evidence of the quantitative advantage of utilizing the FAND filter before performing an analysis on calcium fluorescence data.

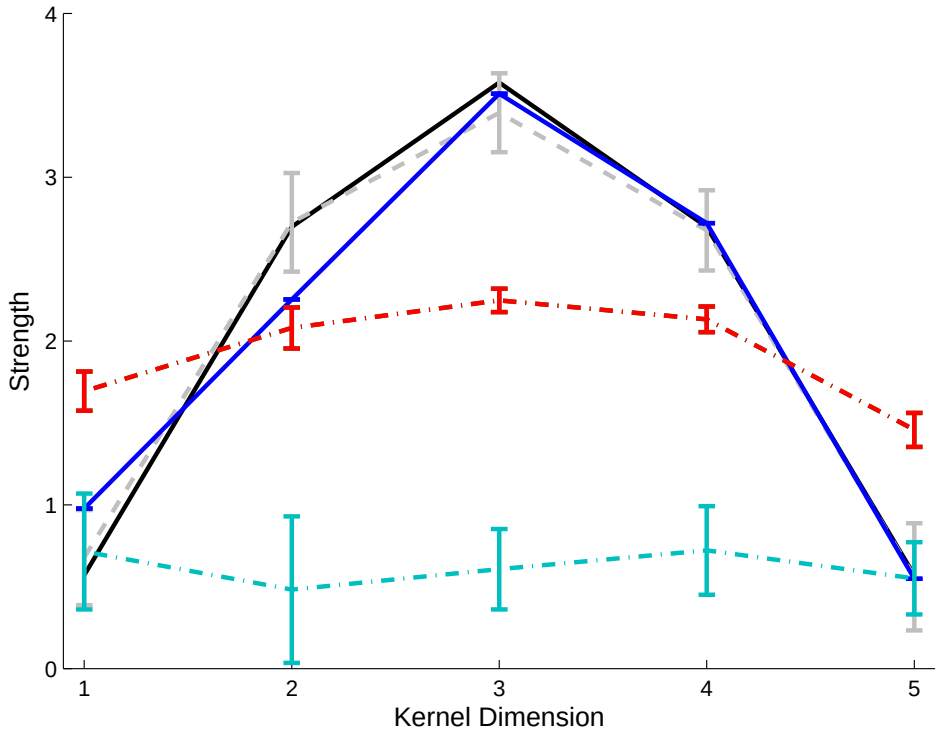

Figure 5: The tuning curve of a neuron (black line), estimating from the true spike train (gray line), and the inferred spike train using the (i) FAND filter (blue line), (ii) Wiener filter (red line), and (iii) dF/F. Clearly, the FAND filter performs nearly as well as the true spike train, whereas the Wiener filter and dF/F do not. Note that half-wave rectification of the Wiener did not change the results at all. Simulation details: mean (solid lines) and standard deviation (bars) of 5 simulations,$T = 800$, $\Delta = 0.005$ sec, $\alpha = 1$, $\beta = 0$, $\tau = 0.1$ sec, $x_{i,t} \sim \mathcal{U}(0, 0.2)$, $\sigma = 0.25$.

3.2 Spatial Filtering

In the previous sections, we implicitly assumed that the raw movie of fluorescence measurements collected by the experimenter had undergone two stages of pre-processing. First, the movie was segmented, to determine regions-of-interest (ROIs). This yields a vector, $\vec{F}_t$, corresponding to the fluorescence intensity at time t for each of the N_p pixels in the ROI. Second, we projected that vector into a scalar, yielding F_t, the assumed input. In this section, we still assume that somebody has gone through our movies and performed some segmentation, but we do not assume that they have projected the vector $\vec{F}_t$ into a scalar F_t^{proj}. Formally, we posit a more general model:

$$\vec{F}_t = \vec{\alpha}(C_t + \beta) + \sigma\vec{\varepsilon}_t, \qquad \vec{\varepsilon}_t \sim \mathcal{N}(\vec{0}, \boldsymbol{I}) \tag{16}$$

where $\vec{F}_t$, $\vec{\alpha}$, $\vec{\varepsilon}_t$, and $\vec{0}$ are all column vectors of length N_p, and $\boldsymbol{I}$ is an $N_p \times N_p$ identity matrix. This model follows because the fluorescence at any individual pixel may be composed of a static element, β, and a dynamic element, that we assume is purely due to calcium fluctuations, C_t. Further, we have assumed that the noise is uncorrelated and has the same variance, σ^2, in each pixel (an assumption that can be relaxed quite easily). Performing inference in this more general model proceeds nearly identical as before:

$$\widehat{\boldsymbol{C}}_z = \underset{\boldsymbol{MC}\geq\boldsymbol{0}}{\operatorname{argmin}} \frac{1}{2\sigma^2} \left\| \vec{\boldsymbol{F}} - \vec{\alpha}(\boldsymbol{C}^\mathsf{T} + \beta\boldsymbol{1}^\mathsf{T}) \right\|^2 + (\boldsymbol{MC})^\mathsf{T}\boldsymbol{\lambda} - z\log(\boldsymbol{MC})^\mathsf{T}\boldsymbol{1}, \tag{17}$$

$$\boldsymbol{g} = -\frac{\boldsymbol{\alpha}}{\sigma^2}(\boldsymbol{F} - \boldsymbol{\alpha}(\widehat{\boldsymbol{C}}_z^\mathsf{T} + \beta)) + \boldsymbol{M}^\mathsf{T}\boldsymbol{\lambda} - z\boldsymbol{M}^\mathsf{T}(\boldsymbol{M}\widehat{\boldsymbol{C}}_z)^{-1} \tag{18}$$

$$\boldsymbol{H} = \frac{\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{\alpha}}{\sigma^2}\boldsymbol{I} + z\boldsymbol{M}^\mathsf{T}(\boldsymbol{M}\widehat{\boldsymbol{C}}_z)^{-2}\boldsymbol{M} \tag{19}$$

Figure 6 demonstrates the utility of this generalization. The top row shows different depictions of an ROI containing a single neuron. On the far left panel is the "true" spatial filter. We modeled the true spatial filter as a sum of Gaussians: a positively weighted small variance Gaussian, and a negatively weighted large variance Gaussian. We chose this model based on our empirical observations that often pixels immediately around a neuron exhibit calcium sensitive fluctuations that are anti-correlated with the pixels on the neuron, probably due to the calcium influx from the vicinity of the neuron. The mean frame (second panel from left) looks very similar to the true spatial filter, as individual frames are effectively just modulating the magnitude of the spatial filter, and adding noise. Typically, to obtain an ROI, one would simply identify the pixels with high positive values from the mean frame, and average them together (third panel from left). Such an approach yields the 1-dimensional fluorescence projection depicted on the left, the typical projection, F_t^{typ}, with its associated inferred spike train beneath. Using the true spatial filter to project $\vec{\boldsymbol{F}}$ onto a 1-dimensional fluorescence time series results on the right, the optimal projection, F_t^{opt}, with its associated inferred spike train beneath. It should be clear that using the true spatial filter improves the SNR of the fluorescence signal, and therefore, the inferred spike train accuracy.

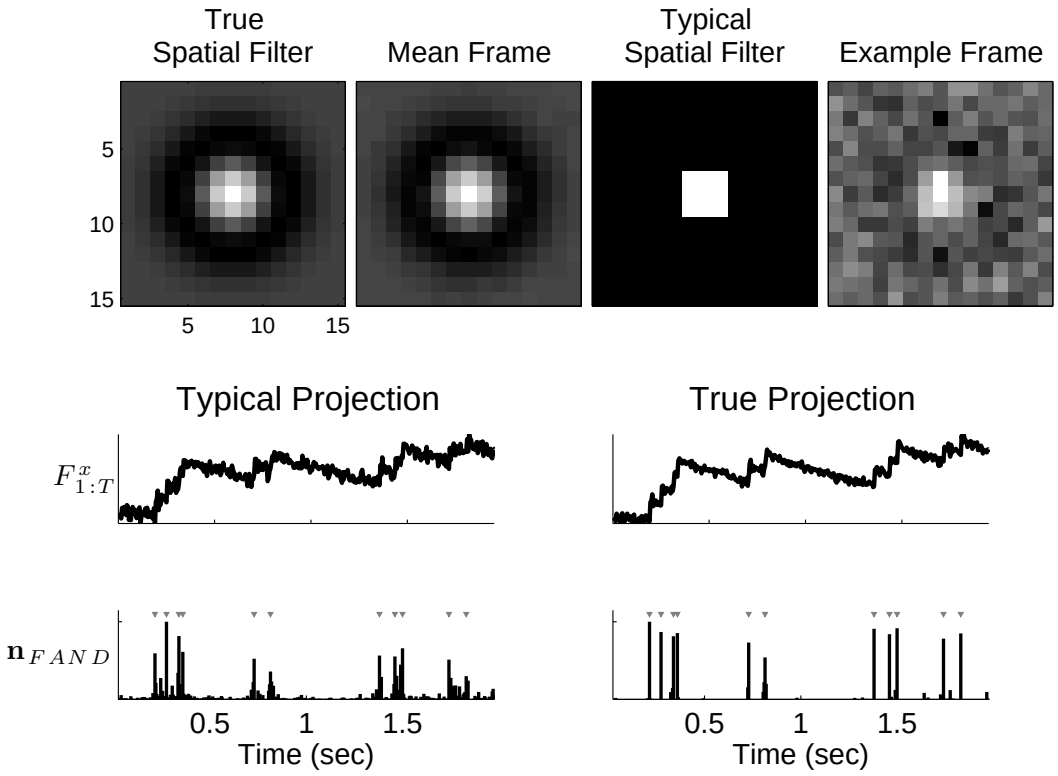

Figure 6: A simulation demonstrating that using a better spatial filter can significantly enhance the effective SNR (see Supplementary Movie 1 for the full movie associated with this simulation). Top left: true spatial filter. Top second from left: mean frame. Top second from right: typical spatial filter. Top right: example frame from move (frame number 100). Middle left: 1-dimensional fluorescence projection using typical spatial filter. Bottom left: $\boldsymbol{n}^{FAND}$ using typical spatial filter. Middle right: 1-dimensional fluorescence projection using true spatial filter. Bottom right: $\boldsymbol{n}^{FAND}$ using true spatial filter. Simulation details: $\boldsymbol{\alpha} = \mathcal{N}(\mathbf{0}, 2\boldsymbol{I}) - 1.1\mathcal{N}(\mathbf{0}, 2.5\boldsymbol{I})$ where $\mathcal{N}(\boldsymbol{mu}, \boldsymbol{\Sigma})$ indicates a Gaussian with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, $\boldsymbol{\beta} = 1$, $\tau = 0.85$ sec, $\lambda = 5$ Hz.

3.3 Learning

In the above, we assumed that the parameters governing our model, $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \beta, \sigma, \gamma, \lambda\} \in \boldsymbol{\Theta}$, were known. In general, however, these parameters must be estimated from the data. Therefore, we take an approximate expectation-maximization approach for computing $\boldsymbol{n}^{FAND}$: initialize some estimate of the parameters, $\widehat{\boldsymbol{\theta}}$, recursively compute $\boldsymbol{n}^{FAND}$ using those parameters and update $\widehat{\boldsymbol{\theta}}$ given $\boldsymbol{n}^{FAND}$, stop recursing when some convergence criteria is met. Below, we provide details for each of the above steps.

Initializing the parameters We initialize our estimate $\boldsymbol{\alpha}$ using Principal Component Analysis (PCA). More specifically, we perform PCA on $\boldsymbol{F}$ (the whole movie), and let $\boldsymbol{\alpha}$ be the PC with the highest eigenvalue. If the movie is very long, we either only use part of it, or simply take the mean frame. Then, we let $\beta = 0$. σ is estimated by finding a sequence of the time series lacking any obvious spikes, and computing the root mean square of that segment (i.e., $\widehat{\sigma} = \sqrt{\vec{F}_{s:t}.^2/(t-s)}$). We typically set γ and λ based on our previous experience with these cells. For instance, $\gamma \approx 0.95$ is often reasonable, and λ is somewhere between 1 and 10 Hz.

Estimating the parameters given $\widehat{\boldsymbol{n}}$ To find the maximum likelihood estimator for the parameters, $\widehat{\boldsymbol{\theta}}$, we must integrate over all possible spike trains. Unfortunately, it is not currently known how to perform this integral exactly, and approximating this integral using Monte Carlo methods is relatively time consuming (see [16] for details). Thus, we resort to a more drastic approximation, commonly used in state-space models. For specifically, instead of integrating over all possibly spike trains, we only consider the most likely sequence (often referred to as the Viterbi path [27]):

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\operatorname{argmax}} \int P[\boldsymbol{F}|\boldsymbol{C},\boldsymbol{\theta}]P[\boldsymbol{C}|\boldsymbol{\theta}]d\boldsymbol{C} \approx \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\operatorname{argmax}} P[\boldsymbol{F}|\widehat{\boldsymbol{C}},\boldsymbol{\theta}]P[\widehat{\boldsymbol{C}},|\boldsymbol{\theta}] \tag{20}$$

where $\widehat{\boldsymbol{C}}$ is determined using the above described inference algorithm. The approximation in (20) is good whenever the likelihood is very peaky, meaning that most of the mass is around the MAP sequence.[2]

Due to the state space nature of the above model (Eqs (16) and (2)), the optimization in Eq (20) simplifies significantly. More specifically, we can write the argument from the right-hand-side of Eq (20) as a product of terms that we have defined in our model:

$$P[\boldsymbol{F}|\widehat{\boldsymbol{C}},\boldsymbol{\theta}]P[\widehat{\boldsymbol{C}}|\boldsymbol{\theta}] = \prod_{t=1}^{T} P[F_t|\widehat{C}_t;\boldsymbol{\alpha},\beta,\sigma]P[\widehat{C}_t|\widehat{C}_{t-1},\widehat{n}_t;\gamma]P[\widehat{n}_t|\lambda]. \tag{21}$$

This optimization simplifies into several separable problems. First, we solve for $\boldsymbol{\alpha}$ and β. Because solving for them jointly is non-concave, we solve for each separately. Consider $\boldsymbol{\alpha}$ only:

$$\widehat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \prod_{t,x=1}^{T,N_p} P_{\boldsymbol{\theta}}[F_{t,x}|C_t] = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \prod_{t,x=1}^{T,N_p} \mathcal{N}(F_{x,t};\alpha_x(C_t+\beta),\sigma^2) \tag{22a}$$

$$= \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \sum_{t,x=1}^{T,N_p} \log\mathcal{N}(F_{x,t};\alpha_x(C_t+\beta),\sigma^2) \tag{22b}$$

$$= \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} -\frac{N_p T}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{t,x=1}^{T,N} (F_{t,x}-\alpha_x(C_t+b))^2 \tag{22c}$$

$$= \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \sum_{t,x=1}^{T,N_p} (F_{t,x}-\alpha_x(C_t+\beta))^2. \tag{22d}$$

Therefore, we can solve for each of the $\widehat{\alpha}_x$'s separately and efficiently using Matlab's `mldivide`: $\widehat{\alpha}_x = (\boldsymbol{C} + \beta\mathbf{1})\backslash\boldsymbol{F}_x$, where $\boldsymbol{F}_x = [F_{1,x},\ldots,F_{T,x}]^\mathsf{T}$. Given $\widehat{\boldsymbol{\alpha}}$, we can estimate β:

[2] The approximation in (20) may be considered a first-order Laplace approximation

$$\widehat{\beta} = \underset{\beta>0}{\operatorname{argmax}} \sum_{t,x=1}^{T,N} (F_{t,x} - \widehat{\alpha}_x(C_t + \beta))^2 = \underset{\beta>0}{\operatorname{argmax}} \sum_{t,x=1}^{T,N} (F_{t,x} - \widehat{\alpha}_x C_t + \beta\widehat{\alpha}_x)^2 \tag{23}$$

which can again be solved efficiently again using Matlab's `mldivide`: $\widetilde{\beta} = \widehat{\alpha}_x \backslash \left(\sum_{t=1}^{T} F_{t,x} - \widehat{\alpha}_x C_t \right)$. If $\widetilde{\beta} < 0$, we simply let $\widehat{\beta} = 0$, else, $\widehat{\beta} = \widetilde{\beta}$.

Given $\widehat{\boldsymbol{\alpha}}$ and $\widehat{\beta}$, we can now estimate σ using the residuals, i.e.,

$$\widehat{\sigma}^2 = \frac{1}{TN_p} \left\| \vec{\boldsymbol{F}} - \widehat{\boldsymbol{\alpha}}(\boldsymbol{C}^{\mathsf{T}} + \beta\mathbf{1}) \right\|^2 \tag{24}$$

Estimating λ is also totally straightforward: $\widehat{\lambda} = \widehat{\boldsymbol{n}}'\mathbf{1}/(T\Delta)$.

We don't update γ as we have found that it does not improve inference quality (not shown), in agreement with previous work [11].

We stop iterating the parameter update whenever (i) iteration number exceeds some upper bound, or (ii) relative change in likelihood does not exceed some lower bound. In practice, we find that parameters tend to converge after several iterations, given our initialization. Figure 7 shows a simulation demonstrating the efficacy of this procedure. The left panel shows the true spatial filter (top), the 1-dimensional fluorescence projection using the true spatial filter, $\boldsymbol{F}^{opt}$ (middle), and the inferred spike train using the true parameters (bottom). The right panel shows the estimated spatial filter, the 1-dimensional fluorescence projection using the estimated filter, $\boldsymbol{F}^{est}$, and the inferred spike train using the estimated filter. Note that we seeded the algorithm with the movie, and parameter estimates that were at least a factor of 2 from the true values.

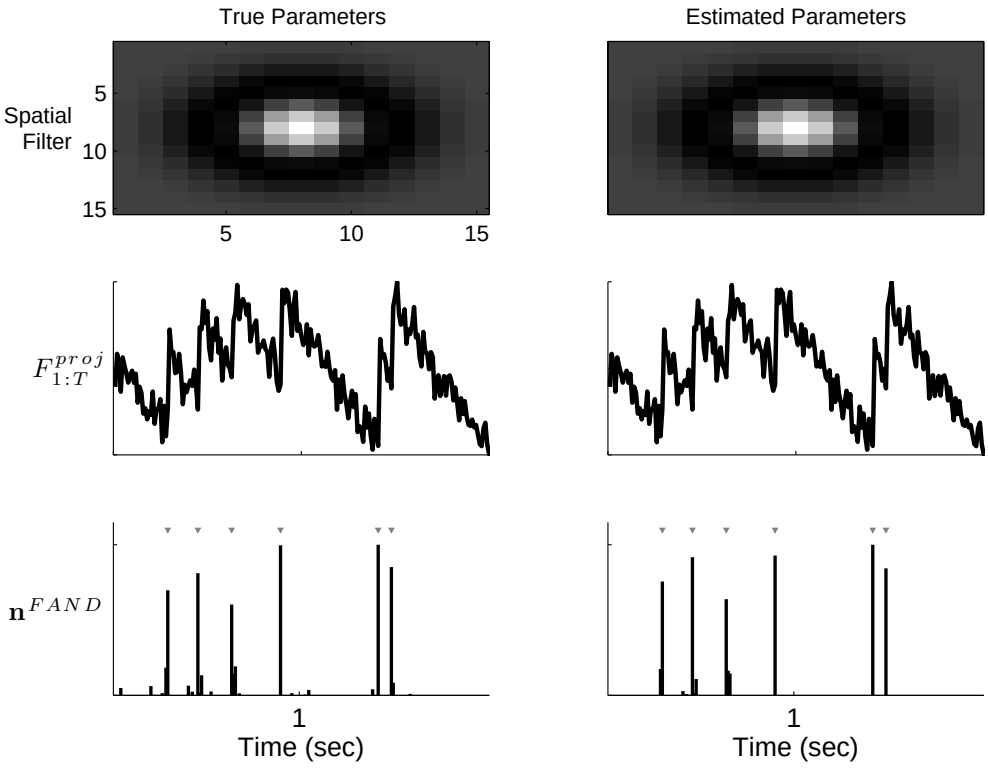

Figure 7: A simulation demonstrating that given only the fluorescence movie, the parameters may be estimated, and the spike train inferred (c.f. Supplementary Movie 2). Top left panel: true spatial filter. Middle left panel: projection of movie onto true spatial filter. Bottom left panel: inferred spike train using true parameters. Right panels: same as left except estimating parameters. All parameters estimated other than γ, which was assumed known. Parameters converged within 7 iterations. Simulation details: $T = 1000$, $\Delta = 5$ msec, $\boldsymbol{\alpha}$ is the same as in Figure 6, $\beta = 0$, $\tau = 500$ msec, $\lambda = 10$ Hz.

3.4 in vitro data

3.5 Overlapping spatial filters

Model

$$\boldsymbol{F}_t = \sum_{i=1}^{N_c} \boldsymbol{\alpha}_i (C_{i,t} + \beta_i) + \boldsymbol{\varepsilon}_t, \qquad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}) \tag{25}$$

$$C_{i,t} = \gamma_i C_{i,t-1} + n_{i,t}, \qquad n_{i,t} \sim \text{Poisson}(n_{i,t}; \lambda_i \Delta) \tag{26}$$

implicit assumption that $\boldsymbol{n}_i \perp \boldsymbol{n}_j$, $\forall i \neq j$

Inference let $\boldsymbol{n} = (n_{1,1}, n_{2,1}, \ldots, n_{N_c,1}, n_{1,2}, \ldots, n_{N_c,T})^\mathsf{T}$. similar def of $\boldsymbol{C}$.

$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & \cdots & & & \\ 1 & -\gamma_1 & 1 & -\gamma_2 & \ldots & 1 & -\gamma_{N_c} & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & & & & \\ 0 & 0 & 0 \ldots & 1 & -\gamma_{N_c-1} & 1 & -\gamma_{N_c} & & \end{bmatrix} \tag{27}$$

inference as before, but replacing the scalar β with $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{N_c})^\mathsf{T}$, and making minor adjustments to deal with dimensionality issues.

Learning estimating $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{N_c})^\mathsf{T}$ is similar. now, we compute $\widehat{\boldsymbol{\alpha}}_x = (\widehat{\alpha}_{1,x}, \ldots, \widehat{\alpha}_{N_c,x})^\mathsf{T}$ using

$$\widehat{\boldsymbol{\alpha}}_x = (\boldsymbol{C} + \widetilde{\boldsymbol{\beta}}) \backslash \boldsymbol{F}_x, \tag{28}$$

where $\widetilde{\boldsymbol{\beta}}$ is $\boldsymbol{\beta}$ reparameterized to be the same size as $\boldsymbol{C}$.

estimating $\boldsymbol{\beta}$ proceeds as before, but since it is a vector, we use Matlab's `quadprog`, imposing the constraint that $\beta_i > 0 \, \forall i$.

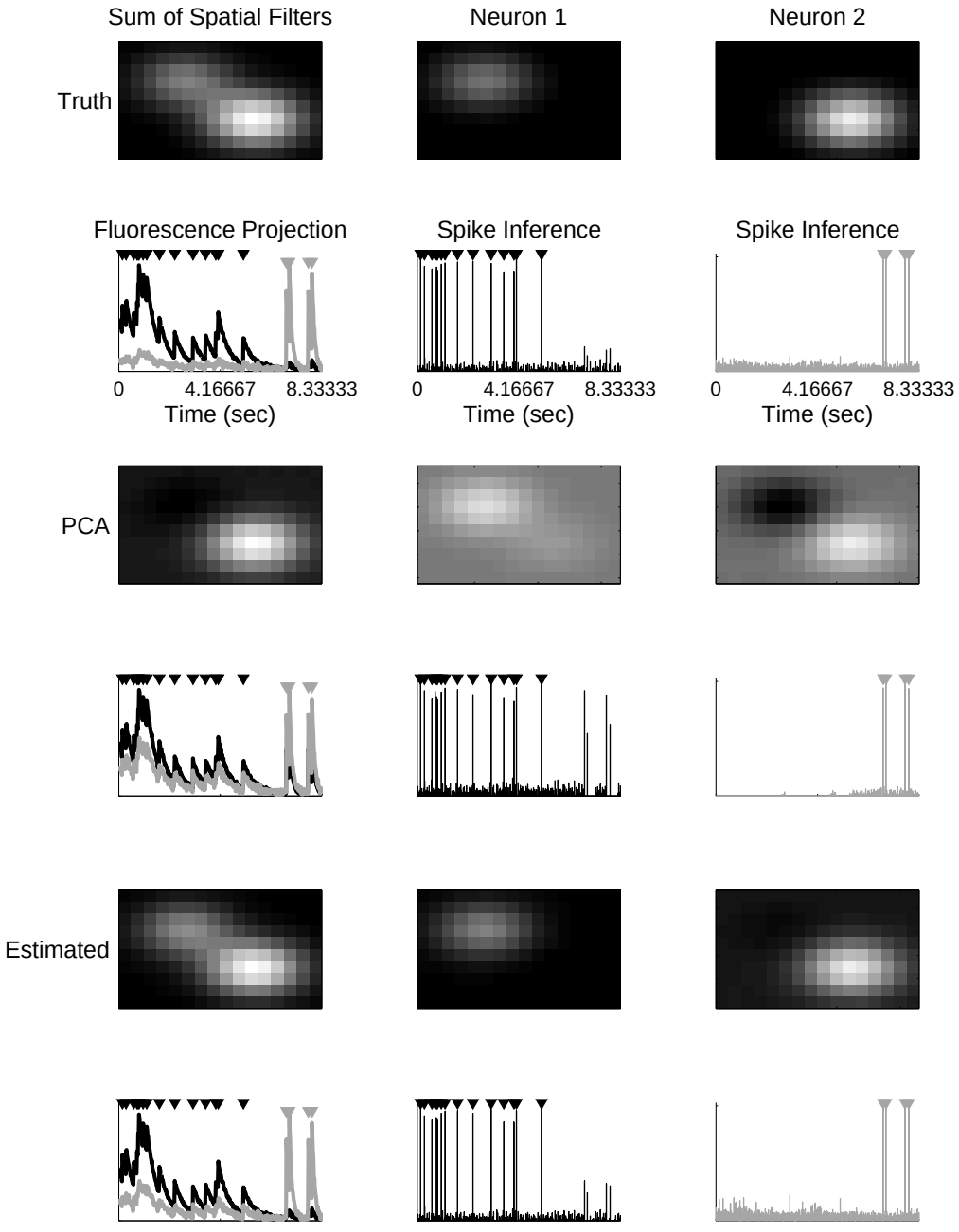

Figure 8: Simulation showing that even when two neuron's spatial filters are largely overlapping

3.6 Population imaging

Figure 9: full movie, in vitro data

3.7 Slow rise time

Figure 10: Slow rise time

3.8 Poisson observation model

Model

$$\boldsymbol{F}_{x,t} \sim \text{Poisson}(\alpha_x(C_t + \beta)) \tag{29}$$

Inference

$$\mathcal{L}_{x,t} = -\alpha_x(C_t + \beta) + F_{x,t}\log(\alpha_x(C_t + \beta)) - \log(F_{x,t}!) \tag{30a}$$
$$g_{x,t} = -\alpha_x + F_{x,t}(C_t + \beta)^{-1} \tag{30b}$$
$$H_{x,t} = -F_{x,t}(C_t + \beta)^{-2} \tag{30c}$$

where $\mathcal{L} = \sum_{x,t} \mathcal{L}_{x,t}$, $\boldsymbol{g} = \sum_x (g_{x,1}, \ldots, g_{x,T})^\mathsf{T}$, and $\boldsymbol{H} = \text{diag}(\sum_x H_{x,t})$.

Figure 11: Poisson

3.9 in vivo data

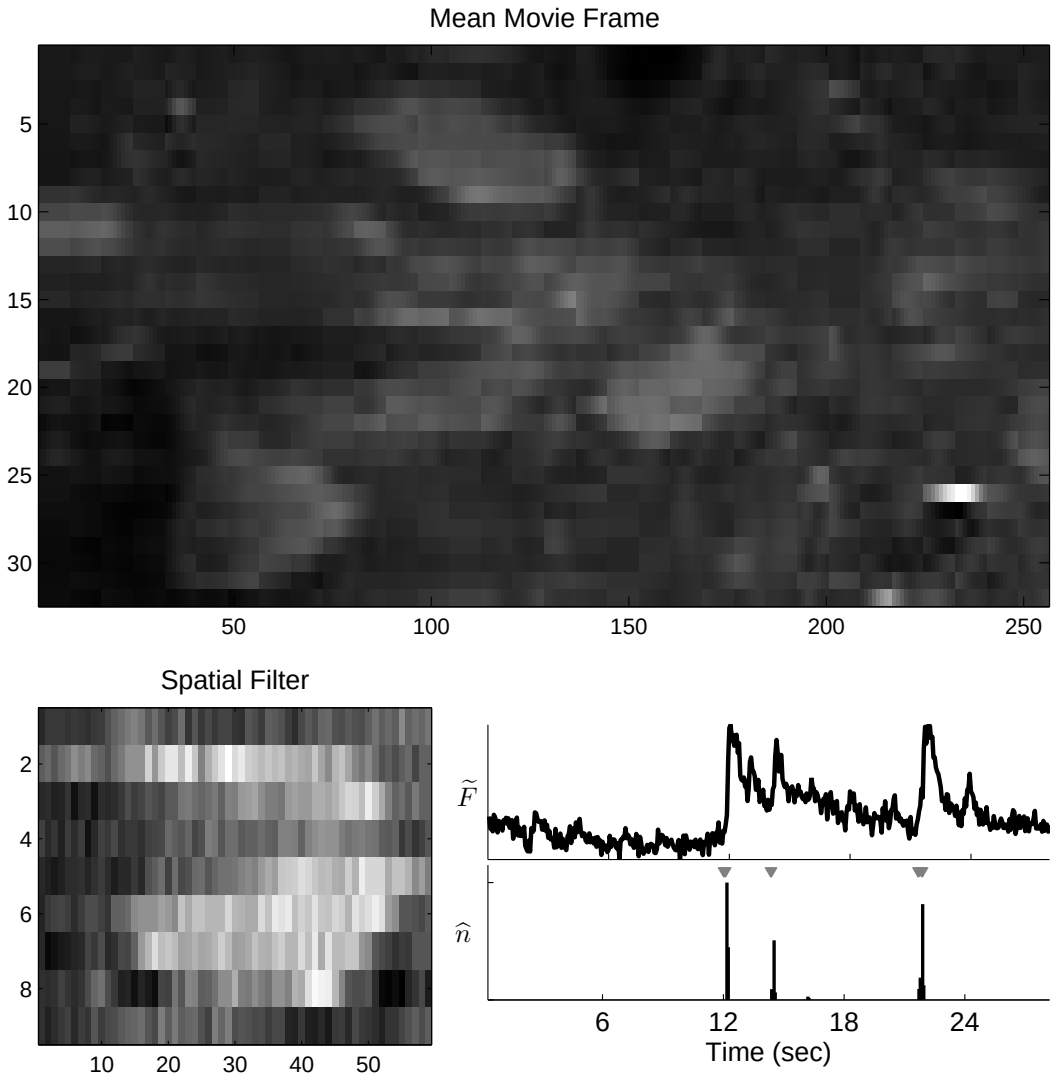

Figure 12: Given only a fluorescence movie, recorded in vivo, we can learn the parameters necessary to correctly infer the spike trains. Left: mean frame. Left: projection of movie onto mean frame. Left: the FAND filter's inference.

4 Discussion

Summary

Extensions

Thresholding

4.1 Incorporating a nonlinear observation model

$$\boldsymbol{F}_t = \boldsymbol{\alpha} S(C_t + \beta) + \boldsymbol{\varepsilon}_t, \qquad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}) \tag{31}$$

where $S(x) = \frac{x^{n_d}}{x^{n_d} + k_d}$

note: initialize with linear result, but add a constant wherever constraint is not satisfied

Figure 13: Saturation

4.2 in vivo data

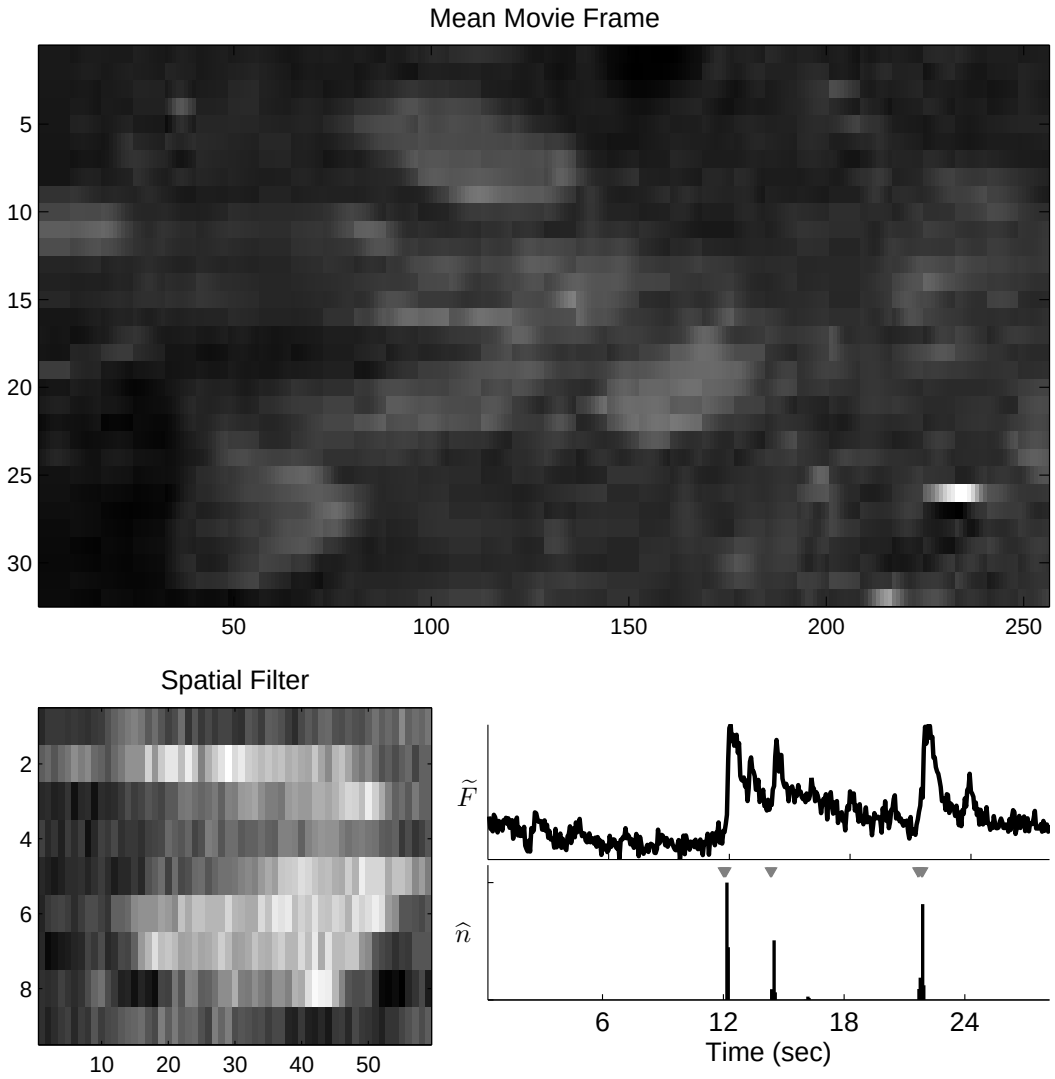

Figure 14: Given only a fluorescence movie, recorded in vivo, we can learn the parameters necessary to correctly infer the spike trains. Left: mean frame. Left: projection of movie onto mean frame. Left: the FAND filter's inference.

4.3 Dynamic prior

Model let $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_T)^\mathsf{T}$

$$C_t = \gamma C_{t-1} + n_t, \qquad n_t \sim \text{Poisson}(n_t; \lambda_t \Delta) \tag{32}$$

Inference

$$\mathcal{L} = \frac{1}{2\sigma^2} \left\| \boldsymbol{F} - \boldsymbol{\alpha}(\boldsymbol{C}^\mathsf{T} + \beta + \boldsymbol{1}^\mathsf{T}) \right\|^2 + (\boldsymbol{MC})^\mathsf{T} \boldsymbol{\lambda} \Delta - z \log(\boldsymbol{MC})^\mathsf{T} \boldsymbol{1} \tag{33a}$$

$$\boldsymbol{g} = -\frac{\boldsymbol{\alpha}}{\sigma^2}(\boldsymbol{F} - \boldsymbol{\alpha}(\widehat{\boldsymbol{C}}_z^\mathsf{T} + \beta)) + \boldsymbol{M}^\mathsf{T} \boldsymbol{\lambda} \Delta - z \boldsymbol{M}^\mathsf{T} (\boldsymbol{M} \widehat{\boldsymbol{C}}_z)^{-1} \tag{33b}$$

Acknowledgments Support for JTV was provided by NIDCD DC00109. LP is supported by an NSF CAREER award, by an Alfred P. Sloan Research Fellowship, and the McKnight Scholar Award. BOW was supported by NDS grant F30 NS051964. The authors would like to thank A. Packer for helpful discussions.

References

[1] R. Yuste and A. Konnerth. *Imaging in Neuroscience and Development, A Laboratory Manual*, 2006.

[2] Shin Nagayama, Shaoqun Zeng, Wenhui Xiong, Max L Fletcher, Arjun V Masurkar, Douglas J Davis, Vincent A Pieribone, and Wei R Chen. In vivo simultaneous tracing and Ca^{2+} imaging of local neuronal circuits. *Neuron*, 53(6):789–803, Mar 2007.

[3] Werner Göbel and Fritjof Helmchen. In vivo calcium imaging of neural network function. *Physiology (Bethesda)*, 22:358–365, Dec 2007.

[4] Liqun Luo, Edward M Callaway, and Karel Svoboda. Genetic dissection of neural circuits. *Neuron*, 57(5):634–660, Mar 2008.

[5] Olga Garaschuk, Oliver Griesbeck, and Arthur Konnerth. Troponin c-based biosensors: a new family of genetically encoded indicators for in vivo calcium imaging in the nervous system. *Cell Calcium*, 42(4-5):351–361, 2007.

[6] Marco Mank and Oliver Griesbeck. Genetically encoded calcium indicators. *Chem Rev*, 108(5):1550–1564, May 2008.

[7] Damian J Wallace, Stephan Meyer zum Alten Borgloh, Simone Astori, Ying Yang, Melanie Bausen, Sebastian Kgler, Amy E Palmer, Roger Y Tsien, Rolf Sprengel, Jason N D Kerr, Winfried Denk, and Mazahir T Hasan. Single-spike detection in vitro and in vivo with a genetic ca2+ sensor. *Nat Methods*, 5(9):797–804, Sep 2008.

[8] Christoph Stosiek, Olga Garaschuk, Knut Holthoff, and Arthur Konnerth. In vivo two-photon calcium imaging of neuronal networks. *Proceedings of The National Academy Of Sciences Of The United States Of America*, 100(12):7319–7324, Jun 2003.

[9] Kenichi Ohki, Sooyoung Chung, Yeang H Ch'ng, Prakash Kara, and R Clay Reid. Functional imaging with cellular resolution reveals precise micro-architecture in visual cortex. *Nature*, 433(7026):597–603, Feb 2005.

[10] Kenichi Ohki, Sooyoung Chung, Prakash Kara, Mark H??bener, Tobias Bonhoeffer, and R. Clay Reid. Highly ordered arrangement of single neurons in orientation pinwheels. *Nature*, 442(7105):925–928, Aug 2006.

[11] Emre Yaksi, Benjamin Judkewitz, and Rainer W Friedrich. Topological reorganization of odor representations in the olfactory bulb. *PLoS Biol*, 5(7):e178, Jul 2007.

[12] Takashi R Sato, Noah W Gray, Zachary F Mainen, and Karel Svoboda. The functional microarchitecture of the mouse barrel cortex. *PLoS Biol*, 5(7):e189, Jul 2007.

[13] J.N.D. Kerr, C.P.J. de Kock, D.S. Greenberg, R.M. Bruno, B. Sakmann, and F. Helmchen. Spatial organization of neuronal population responses in layer 2/3 of rat barrel cortex. *Journal of Neuroscience*, 27(48):13316, 2007.

[14] Ilker Ozden, H. Megan Lee, Megan R Sullivan, and Samuel S-H Wang. Identification and clustering of event patterns from in vivo multiphoton optical recordings of neuronal ensembles. *J Neurophysiol*, 100(1):495–503, Jul 2008.

[15] David S Greenberg, Arthur R Houweling, and Jason N D Kerr. Population imaging of ongoing neuronal activity in the visual cortex of awake rats. *Nat Neurosci*, 11(7):749 – 751, Jun 2008.

[16] JT Vogelstein, BO Watson, Packer AM, R Yuste, Jedynak B, and L Paninski. Spike inference from calcium imaging using sequential monte carlo methods. *Biophysical Journal*, 2009.

[17] Terrence F Holekamp, Diwakar Turaga, and Timothy E Holy. Fast three-dimensional fluorescence imaging of activity in neural populations by objective-coupled planar illumination microscopy. *Neuron*, 57(5):661–672, Mar 2008.

[18] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, Oct 1999.

[19] D.D. Lee and H.S. Seung. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13:556–562, 2001.

[20] Quentin J M Huys, Misha B Ahrens, and Liam Paninski. Efficient estimation of detailed single-neuron models. *J Neurophysiol*, 96(2):872–890, Aug 2006.

[21] L.F. Portugal, J.J. Judice, and L.N. Vicente. A comparison of block pivoting and interior-point algorithms for linear least squares problems with nonnegative variables. *Mathematics of Computation*, 63(208):625–643, 1994.

[22] H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. *Advances in Neural Information Processing Systems*, 2006.

[23] O?Grady, P.D. and Pearlmutter, B.A. Convolutive non-negative matrix factorisation with a sparseness constraint. *Machine Learning for Signal Processing, 2006. Proceedings of the 2006 16th IEEE Signal Processing Society Workshop on*, pages 427–432, 2006.

[24] N. Wiener. *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*. MIT Press, Cambridge, Mass., 1949.

[25] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[26] S. Boyd and L. Vandenberghe. *Convex Optimization*. Oxford University Press, 2004.

[27] Lawrence R Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 72(2):257–286, February 1989.

A Wiener Filter

Sections 2.3 outline one approach to solving Eq. (3), by approximating the Poisson distribution with an exponential distribution, and imposing a non-negative constraint on the inferred $\widehat{\boldsymbol{n}}$. Perhaps a more straightforward approach would be to approximate the Poission distribution with a Gaussian distribution. In fact, as rate increases above about 10 spikes/sec, a Poisson distribution with rate $\lambda\Delta$ is well approximated by a Gaussian with mean and variance $\lambda\Delta$. Given such an approximation, instead of Eq. (6), we would obtain:

$$\widehat{\boldsymbol{n}}_w \approx \underset{n_t \in \mathbb{R}, \forall t}{\operatorname{argmin}} \sum_{t=1}^{T} \left(\frac{1}{2\sigma^2}(F_t - C_t)^2 + \frac{1}{2\lambda\Delta}(n_t - \lambda\Delta)^2 \right) \tag{34}$$

As above, we can rewrite Eq. (34) in matrix notation in terms of $\boldsymbol{C}$:

$$\widehat{\boldsymbol{C}}_w = \underset{C_t \in \mathbb{R}, \forall t}{\operatorname{argmin}} \frac{1}{2\sigma^2} \|\boldsymbol{F} - \boldsymbol{C}\|^2 + \frac{1}{2\lambda\Delta} \|\boldsymbol{M}\boldsymbol{C} - \lambda\Delta\boldsymbol{1}\|^2 \tag{35}$$

which is quadratic in $\boldsymbol{C}$, and may therefore be solved analytically using quadratic programming, $\widehat{\boldsymbol{C}}_w = \widehat{\boldsymbol{C}}_0 + \boldsymbol{d}_w$, where $\widehat{\boldsymbol{C}}_0$ is the initial guess and $\boldsymbol{d}_w = \boldsymbol{H}_w \backslash \boldsymbol{g}_w$, where

$$\boldsymbol{g}_w = \frac{1}{\sigma^2}(\boldsymbol{C}_0' - \boldsymbol{F}) + \frac{1}{\lambda\Delta}((\boldsymbol{M}\widehat{\boldsymbol{C}}_0)'\boldsymbol{M} - \lambda\Delta\boldsymbol{M}'\boldsymbol{1}) \tag{36}$$

$$\boldsymbol{H}_w = \frac{1}{\sigma^2}\boldsymbol{I} + \frac{1}{\lambda\Delta}\boldsymbol{M}'\boldsymbol{M} \tag{37}$$

Note that this solution is the optimal linear solution, under the assumption that spikes follow a Gaussian distribution, and if often referred to as the Wiener filter, regression with a smoothing prior, or ridge regression. To estimate the parameters for the Wiener filter, we take the same approach as above:

$$\widehat{\boldsymbol{\theta}}_w \approx \underset{\boldsymbol{\theta}_w}{\operatorname{argmax}} P[\boldsymbol{F}|\widehat{\boldsymbol{n}}_w, \boldsymbol{\theta}_w) P[\widehat{\boldsymbol{n}}_w|\boldsymbol{\theta}_w) \tag{38a}$$

$$= \underset{\boldsymbol{\theta}_w}{\operatorname{argmax}} -\frac{T}{2}\log(4\pi^2\sigma^2\lambda\Delta) - \frac{1}{2\sigma^2}\|\boldsymbol{Y}_w + \boldsymbol{\eta}_w\boldsymbol{X}_w\|^2 - \frac{1}{2\lambda\Delta}\|\widehat{\boldsymbol{n}}_w - \lambda\Delta\boldsymbol{1}\|^2 \tag{38b}$$

where $\boldsymbol{Y}_w$, $\boldsymbol{\eta}_w$, and $\boldsymbol{X}_w$ are defined as their subscriptless counterparts in Eq. (20).