Advances in Statistical Machine Learning Methods for Neural Data Science

Ding Zhou

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2021

Abstract

Advances in Statistical Machine Learning Methods for Neural Data Science

Ding Zhou

Innovations in neural data recording techniques are revolutionizing neuroscience and presenting both challenges and opportunities for statistical data analysis. This dissertation discusses several recent advances in neural data signal processing, encoding, decoding, and dimension reduction. Chapter 1 introduces challenges in neural data science and common statistical methods used to address them. Chapter 2 develops a new method to detect neurons and extract signals from noisy calcium imaging data with irregular neuron shapes. Chapter 3 introduces a novel probabilistic framework for modeling deconvolved calcium traces. Chapter 4 proposes an improved Bayesian nonparametric extension of the hidden Markov model (HMM) that separates the strength of the self-persistence prior and transition prior. Chapter 5 introduces a more identifiable and interpretable latent variable model for Poisson observations. We develop efficient algorithms to fit each of the aforementioned methods and demonstrate their effectiveness on both simulated and real data.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I first would like to thank my advisor Prof. Liam Paninski. I appreciate his brilliant ideas and guidance on my research, as well as his continuous encouragement for me to talk with more people. I learned from him how to think independently and complete a research project.

I have been incredibly lucky to work in an amazing group. I enjoy the academic atmosphere in our group. I would like to acknowledge my friends and collaborators: Dr. Yuanjun Gao, for your support and company; Prof. Xuexin Wei, for your insights and many delicious foods; Prof. Pengcheng Zhou, for your enthusiasm and for opening me the door of this great group. It has been my pleasure collaborating with them on various exciting projects. I also benefit a lot from talking with my excellent group members. I would like to thank Dr. Kenneth Kay, Ian Kinsella, Dr. Matthew Whiteway, Dr. Ari Pakman, Prof. Xinyi Deng, Dr. Joshua Glaser, among others, for their generous help and fruitful conversations.

I would like to thank my thesis committee: Prof. Tian Zheng, Prof. John Cunningham, Prof. John Paisley, and Prof. Xuexin Wei, for their constructive suggestions. Prof. Tian Zheng and Prof. John Cunningham also served on my oral exam committee and gave me useful comments. Prof. John Paisley gave me helpful advice on my HMM project.

I would also like to thank Prof. Jingchen Liu and Prof. Zhiliang Ying for their guidance and support in my early Ph.D. stage. I am grateful to the department administrators Dood Kalicharan and Anthony Cruz for always being there for me.

I appreciate my mentors: Dr. Li Pan, Dr. Xu Gao, Dr. Tianwen Chen, Dr. Scott Rome,

and Dr. Hongcheng Wang, for their help during and after my summer internships at Google and Comcast Applied AI Lab. These two enjoyable summer internships have broadened my views of many interesting statistics questions in the real world.

I am grateful to my friends Owen Ward, Chengliang Tang, Alejandro Quintos Lima, Bridget Ratcliffe, Dr. Haochen Xu, Dr. Siliang Zhang, Dr. Guanhua Fang, Zhi Wang, Chaoyu Yuan, Gengyu Guo, Xinlu Xiao, Fangyuan Chang, among others, for their kind help and support throughout my Ph.D. journey.

Finally, I would like to thank my parents for their patience, support, encouragement, and unconditional love. This Ph.D. would not be possible without them.

To my parents.

Chapter 1: Introduction and background

1.1 Neural data science

Innovations in large-scale multi-neuronal data recording (e.g. electrophysiological recordings and calcium imaging) are revolutionizing neuroscience. Captured recordings are composed of noisy observations of *neural spike trains* for multiple neurons. Each neural spike train is a time series of discrete events at which a neuron fires. The size of noisy recorded data poses both challenges and opportunities for subsequent statistical data analysis.

1.1.1 Recordings

Electrophysiological recordings simultaneously record electrical signals of large neural populations by inserting multiple electrodes into animals' brain tissues. Each electrode records the electrical activities of several nearby neurons. Calcium imaging is another widely-used technique for recording activities of large neural populations simultaneously at single-neuron resolution [1, 2, 3, 4]. The idea behind calcium imaging is that it provides a slow, nonlinear encoding of underlying spike trains through the fluorescence emitted by calcium indicators. Compared to electrophysiological recordings, calcium imaging has several important advantages [5]: it offers high spatial resolution, can achieve cell-type specificity, and has been proven to be scalable to record up to $O(10^4)$ neurons simultaneously in vivo (which is over an order of magnitude more than electrophysiological recordings). However, this high spatial resolution and large scale come at a cost since calcium imaging records neural spikes with lower temporal precision than electrophysiological recordings.

1.1.2 Signal pre-processing

Neural spike trains encode external stimuli and behavioral variables that the brain processes. To understand how the brain works, scientists must first extract the spike train signals of interest from recordings. Modern neural data recording techniques produce datasets with hundreds or even thousands of neurons, which requires reliable, automated signal preprocessing methods for these recordings.

In electrophysiological recordings, we need to detect spiking activities within the signals recorded by each electrode and assign these spikes to individual neurons responsible for them. This pre-processing step is called spike sorting. State-of-the-art spike sorting pipelines apply clustering methods to obtain waveform templates for each recorded neuron first [6, 7] followed by a deconvolution step with these waveform templates to infer spike times for each neuron [8, 7]. See [9, 10] for the detailed review of these methods.

For calcium imaging, it is standard to decompose recordings into a library of spatial shapes and temporal calcium traces corresponding to the imaged neurons. This preprocessing procedure is known as demixing, for which matrix factorization based methods are commonly applied [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. Neurons with prominent axons and dendrites have irregular shapes that traditional demixing methods struggle to effectively extract. Chapter 2 develops a novel demixing method, which extends these techniques to be especially useful for axonal and dendritic imaging data.

Once neuron shapes and fluorescence traces are extracted from raw calcium data, scientists often wish to infer the underlying spike train for each neuron. There is an expansive literature focusing on methods which deconvolve the fluorescence trace in order to infer the spiking activity [22, 23, 14, 24, 25, 26, 27, 28, 29, 30, 31, 32].

1.1.3 Neural encoding and decoding

Neural encoding characterizes how spike trains will change when varying external inputs (e.g. stimuli or behavioral variables), and neural decoding considers the reverse problem of

predicting external inputs from neural population responses. Traditionally, generalized linear models (GLMs) and Bayesian posterior inference are applied to perform neural encoding and decoding analyses [33, 34]. More recently, nonlinear regression models have been applied to solve the neural encoding and decoding problem [35, 36]. These models express neural firing rates as a function of external inputs, which is the encoding model; and they can be inverted to decode external inputs from neural activities through Bayesian analysis.

Although there exist intensive analyses for neural spike trains from electrophysiological recordings, the analysis for deconvolved calcium "spikes" receives much less attention. Consequently, there does not yet exist a "standard model" for deconvolved "spikes". Chapter 3 introduces a new probabilistic framework to fill in this gap.

1.2 Dimension reduction methods for neural data

Large-scale neural data is usually high-dimensional, presenting an opportunity for research focused on extracting meaningful low-dimensional signals which underlie neural activities. Statistical dimension reduction methods are widely used to find structures in neural data. See [37] for a perspective.

1.2.1 Linear dimension reduction

Linear dimension reduction methods factorize the data X in the following way,

$$X \approx Z \cdot L, \tag{1.1}$$

where $X \in \mathbb{R}^{T\times p}, Z \in \mathbb{R}^{T\times k}, L \in \mathbb{R}^{k\times p}, k \ll p$. Here T is the number of samples, p is the dimension of high-dimensional observations, and k is the dimension of low-dimensional factors. Note that for neural spike trains, T is the number of time steps, and p is the number of neurons.

The most commonly used dimension reduction methods are principal component analysis

(PCA) and more-general factor analysis (FA). These methods factorize the data X into a product of low-dimensional factors Z which preserve the variation of the data as much as possible and loadings L which represent weights of the data on each factor. See applications of PCA and FA to neural spike trains in [38, 39].

Independent component analysis (ICA) [40] linearly transforms the data into maximally independent factors. For an application of ICA to calcium imaging demixing see [11].

Sometimes it is necessary to impose constraints on factors and loadings to model properties of the data. A popular choice of constraint is to impose nonnegativity on factors and/or loadings, and the method for doing so is called nonnegative matrix factorization (NMF). NMF is commonly used for imaging data analysis because the imaging data is nonnegative and meaningful factors and loadings should also be nonnegative. One natural application of NMF is to calcium imaging demixing [13, 14, 15, 18, 19, 20].

Another linear dimension reduction method for the data with labels is linear discriminant analysis (LDA) [41], which maps the data to the direction which separates different classes most. [42, 43] provide examples of applying LDA to neural spike trains.

1.2.2 Nonlinear dimension reduction

Nonlinear dimension reduction methods may outperform linear dimension reduction methods by finding more complicated structures within the data.

Autoencoders (AEs) [44] map the data into a smaller space that maintains distance between dissimilar points and preserves significant global structures of the original data. An AE first projects the original data X into the low-dimensional space through the nonlinear encoder $\mathrm{f} : \mathbb{R}^p \to \mathbb{R}^k$, then reconstructs X by projecting the low-dimensional data back to the original space through another nonlinear decoder $\mathrm{g} : \mathbb{R}^k \to \mathbb{R}^p$. Both f and g are

parameterized by feed-forward neural networks. The model is defined as

$$\begin{aligned} \text{encoder: } Z &= \mathrm{f}(X), \\ \text{decoder: } X^{'} &= \mathrm{g}(Z), \end{aligned} \tag{1.2}$$

where Z consists of low-dimensional factors, $X^{'}$ is the reconstructed data, and an AE learns f, g together by minimizing the Euclidean distance between the original and reconstructed data. [45] provides an example of applying an AE to cortical neural populations.

Variational autoencoders (VAEs) [46, 47] generalize AEs to probabilistic encoders and decoders. A VAE is specified as

$$\begin{aligned} Z_t &\sim p_\theta(z), \\ \text{probabilistic decoder} &: X_t \sim p_\theta(x|Z_t),\ t = 1 \cdots T, \\ \text{probabilistic encoder} &: q_\phi(Z_t|X_t), \end{aligned} \tag{1.3}$$

where $Z_t \in \mathbb{R}^k$ is an unobserved random variable (i.e. a latent variable) which is independently generated from the prior distribution $p_\theta(z)$. The t-th observation $X_t \in \mathbb{R}^p$ is assumed to be generated from the conditional distribution $p_\theta(x|Z_t)$. Here θ are parameters of the generative model. For example, the latent prior $p_\theta(z)$ can be specified as $\mathcal{N}\left(\mu_0, \mathrm{diag}\left(\sigma_0^2\right)\right)$, where the mean μ_0 and standard deviation σ_0 are both in $\mathbb{R}^k$. The conditional distribution $p_\theta(x|Z_t)$ can be defined as $\mathcal{N}\left(\mathrm{g}\left(Z_t\right), \mathrm{diag}\left(\psi^2\left(Z_t\right)\right)\right)$, where g, ψ are parameterized using feed-forward neural networks from $\mathbb{R}^k \to \mathbb{R}^p$, and determine the mean and standard deviation of the conditional distribution respectively. The conditional distribution $p(x|Z_t)$ is also referred to as a probabilistic decoder. To perform inference, the intractable true posterior $p(Z_t|X_t)$ is approximated with the recognition model $q_\phi(Z_t|X_t)$, which is also parameterized by feed-forward neural networks. This is also known as a probabilistic encoder. More details are shown in the Inference section 1.3.2 next.

Nonlinear latent variable models, such as VAEs, are expressive and flexible, so they are capable of extracting complex nonlinear structures which would be lost to linear methods. However, VAEs are often unidentifiable as shown in [48, 49]. Inferred latent structures can differ from the ground truth up to random nonlinear transformations; hence, latent variables these models extract usually lack interpretability. Previously, some efforts have been made to encourage disentanglement and enhance model interpretability [50, 51, 52]. However, none of these methods offer a theoretical guarantee of identifiability.

Recently, some literature has been developed to address this identifiability issue [49, 53, 54, 55]. Chapter 5 generalizes a recently proposed identifiable variational autoencoder (iVAE) [49, 53] to Poisson observations and applies it to analyze high-dimensional neural spike trains. The proposed method integrates external inputs (e.g. stimuli or behavioral variables) in generalized linear models and latent variable models, together achieving model identifiability and improved interpretability.

Another class of nonlinear methods, such as Isomap [56] and locally-linear embedding (LLE) [57], maps similar data points closer together thus preserving local structures of the high dimensional data. See applications of Isomap in [58] and LLE in [59, 60] to neural spike trains.

The third class of nonlinear methods preserves both global and local structures of the data, e.g. t-SNE [61] and UMAP [62]. These two methods learn low-dimensional representations with the structure as similar as possible to the high-dimensional data. There are parameters in these two methods to adjust the balance between the global structure and local structure. See [63] for an application of t-SNE to spike sorting.

1.3 Latent variable models for time-series data dimension reduction

Neural spike trains usually exhibit strong temporal correlation; therefore, modeling temporal structures of neural spike trains is important. Regression-based methods (e.g. generalized linear models for the neural encoding and decoding problem) directly input stimuli or

behavioral variables to the model and assume that observed neural spike trains are temporally independent given those inputs. However, sometimes we believe that there exist other unobserved variables which may affect neural activities. In this case, both latent variables and model parameters need to be estimated from the data. These latent variable models further provide flexibility for dimension reduction by assuming that low-dimensional representations have some temporally dependent distributions.

1.3.1 Models

In general, these latent variable models assume that observed high-dimensional neural activities are modulated by underlying unobserved low-dimensional latent variables, and latent variables are usually assumed to follow a Markov process. Two commonly used latent variable models with a Markov process prior are the hidden Markov model (HMM) [64] and state-space model (SSM) [65].

The HMM is specified as follows:

$$\begin{aligned} Z_t &\sim \text{multinomial}\left(\pi_{Z_{t-1}}\right), \\ X_t &\sim p\left(x|\theta_{Z_t}\right), \end{aligned} \tag{1.4}$$

where $X_t \in \mathbb{R}^p, Z_t \in \mathbb{R}^k, k \ll p$ are the observed and latent variables at time step t respectively. The latent state Z_t follows a multinomial distribution with the probability mass function π depending on Z_{t-1}. Given Z_t, the observation X_t is independent from previous latent states $Z_{1:t-1}$ and follows the probability density function p with the parameters θ determined by Z_t. The probability density function p is usually chosen as a multinomial or Gaussian distribution. For neural spike trains, it is usually switched to a Poisson distribution [66, 67]. Instead of fixing the number of latent states as in the parametric HMM, Bayesian nonparametric HMMs, e.g. the hierarchical Dirichlet process HMM (HDP-HMM) [68], assume that there is an infinite number of latent states and can automatically learn the number of states

from the data. In Chapter 4, we present a detailed introduction to the HDP-HMM and propose an improved method called the disentangled sticky HDP-HMM (DS-HDP-HMM), which separates the strength of the self-persistence prior and transition prior.

With similar notation as the HMM, the SSM can be defined as follows:

$$\begin{aligned} Z_t &\sim \mathcal{N}\left(AZ_{t-1} + b, Q\right), \\ X_t &\sim \mathcal{N}\left(CZ_t + d, \Sigma\right), \end{aligned} \tag{1.5}$$

where Z_t follows an autoregressive process of order 1 (AR(1) process), and X_t is a linear transformation of Z_t with Gaussian noises. The evolution matrix $A \in \mathbb{R}^{k\times k}$ sets the linear dynamics of Z_t, while the coefficient matrix $C \in \mathbb{R}^{k\times p}$ controls the dependence of X_t on Z_t. The intercepts $b \in \mathbb{R}^k$ and $d \in \mathbb{R}^p$ control the long-term mean of Z_t and X_t respectively, while the covariance matrices $Q \in \mathbb{R}^{k\times k}$ and $\Sigma \in \mathbb{R}^{p\times p}$ set the noise level of Z_t and X_t respectively. This model is well known as the classic Kalman filter [65]. There are many variants of this simple linear SSM. For example, when modeling neural spike trains, the conditional distribution of X_t given Z_t is usually switched to a Poisson distribution [69, 70, 71, 72, 73]. The linear transformations in the latent prior and conditional distribution can both extend to nonlinear functions. This extension can be viewed as a connection between the VAE and SSM [71, 72, 74].

1.3.2 Inference

The expectation-maximization (EM) algorithm [75] is the traditional inference method for the HMM and Kalman filter models. However, in many cases, e.g. an SSM with Poisson observations, the posterior $p(Z|X)$ is intractable, so there is no analytical solution for the EM algorithm. When this happens, approximate inference methods are required for inference.

One class of approximate inference methods is the Markov chain Monte Carlo (MCMC) sampling. This class contains classical methods such as the Metropolis-Hastings algo-

rithm [76, 77] and Gibbs sampler [78] which have been widely used to fit Bayesian models; see [79] for a detailed review. The MCMC sampling constructs a Markov Chain whose stationary distribution is the true posterior $p(Z|X)$. Upon convergence, the MCMC sampling generates exact samples from the target posterior, and we can use these samples to approximate the posterior. In practice, it suffers from a slow convergence rate and is hard to scale to large data [79].

Variational inference (VI) [80, 81] has been proposed as an alternative method to MCMC. The VI algorithm approximates the posterior with a simpler distribution family, such as Gaussian distributions, known as the variational family. The algorithm finds the candidate distribution closest to the posterior within the variational family by maximizing a lower bound of the evidence $\log p(X)$. The objective function is called the evidence lower bound (ELBO), and is defined as

$$\text{ELBO}(q) = \mathbb{E}_q[\log p(X, Z) - \log q(Z)], \tag{1.6}$$

where q is the candidate approximate distribution. The VI algorithm takes advantage of optimization methods such as stochastic optimization [82] and is, therefore, computationally faster than the MCMC sampling. As a trade-off, the VI algorithm can only find an approximation to rather than the exact true posterior.

However, the VI algorithm still requires that the expectation with respect to the approximate distribution in the ELBO equation (1.6) has desirable analytical solutions, which is intractable in many cases, for example, when the link function f is a feed-forward neural network as in a VAE. The auto-encoding variational Bayes (AEVB) algorithm [46] has been proposed to perform inference for these intractable cases. The AEVB algorithm models the approximate distribution as a probabilistic encoder, and uses a reparametrization trick in the ELBO equation (1.6) to optimize it through stochastic gradient descent. It is generally faster and more scalable than the VI algorithm.

For the latent variable models proposed in Chapter 4 and 5, we develop efficient MCMC and AEVB algorithms respectively to fit them.

1.4 Dissertation outline

A brief overview of the rest of the dissertation is as follows.

Chapter 2 relates to [20, 83], which develops a new demixing method for calcium imaging. This includes a fast initialization and subsequent constrained NMF approach that is especially effective on axonal and dendritic imaging data. We apply this new demixing method to two challenging applications: dendritic calcium imaging data and voltage imaging data in the context of optogenetic stimulation. In both cases, we show that our new approach leads to better extraction of activity from the video data.

Chapter 3 corresponds to [84], which introduces a novel probabilistic framework for the modeling of deconvolved calcium traces. Much effort has been devoted to developing "pre-processing" tools for calcium video data, addressing the important issues of e.g., motion correction, denoising, compression, demixing, and deconvolution. However, statistical modeling of deconvolved calcium signals (i.e., the estimated activity extracted by a pre-processing pipeline) is just as critical for interpreting calcium measurements, and for incorporating these observations into downstream probabilistic encoding and decoding models. Surprisingly, these issues have to date received significantly less attention. In this work, we examine the statistical properties of the deconvolved activity estimates and compare probabilistic models for these random signals. In particular, we propose a zero-inflated gamma (ZIG) model, which characterizes the calcium responses as a mixture of a gamma distribution and a point mass that serves to model zero responses. We apply the resulting models to neural encoding and decoding problems. We find that the ZIG model outperforms simpler models (e.g., Poisson or Bernoulli models) in the context of both simulated and real neural data, and can therefore play a useful role in bridging calcium imaging analysis methods with tools for analyzing activity in large neural populations.

Chapter 4 corresponds to [85], which develops an improved Bayesian nonparametric HMM. The hierarchical Dirichlet process hidden Markov model (HDP-HMM) has been used widely as a natural Bayesian nonparametric extension of the classical Hidden Markov Model for learning from sequential and time-series data. A sticky extension of the HDP-HMM has been proposed to strengthen the self-persistence probability in the HDP-HMM. However, the sticky HDP-HMM entangles the strength of the self-persistence prior and transition prior together, limiting its expressiveness. Here, we propose a more general model: the disentangled sticky HDP-HMM (DS-HDP-HMM). We develop novel Gibbs sampling algorithms for efficient inference in this model. We show that the disentangled sticky HDP-HMM outperforms the sticky HDP-HMM and HDP-HMM on both synthetic and real data, and apply the new approach to analyze neural data and segment behavioral video data.

Chapter 5 corresponds to [86], which introduces a more identifiable and interpretable latent variable model for Poisson observations. Recently, deep generative models have been proposed to fit neural population responses. While these methods are flexible and expressive, the downside is that they can be difficult to interpret and identify. To address this problem, we propose a method that integrates key ingredients from latent models and traditional neural encoding models. Our method, pi-VAE, is inspired by recent progress on identifiable variational autoencoder, which we adapt to make appropriate for neuroscience applications. Specifically, we propose to construct latent variable models of neural activity *while simultaneously* modeling the relation between the latent and task variables (non-neural variables, e.g. sensory, motor, and other externally observable states). The incorporation of task variables results in models that are not only more constrained, but also show qualitative improvements in interpretability and identifiability. We validate pi-VAE using synthetic data and apply it to analyze neurophysiological datasets from rat hippocampus and macaque motor cortex. We demonstrate that pi-VAE not only fits the data better but also provides unexpected novel insights into the structure of the neural codes.

Chapter 2: Demixing for calcium imaging data

This chapter was included as a part of a larger joint work "Penalized matrix decomposition for denoising, compression, and improved demixing of functional imaging data" [20] in collaboration with E. Kelly Buchanan, Ian Kinsella, Rong Zhu, Pengcheng Zhou, Felipe Gerhard, John Ferrante, Ying Ma, Sharon H. Kim, Mohammed A Shaik, Yajie Liang, Rongwen Lu, Jacob Reimer, Paul G Fahey, Taliah N Muhammad, Graham Dempsey, Elizabeth Hillman, Na Ji, Andreas S Tolias, and Liam Paninski. We thank Darcy Peterka and Jack Bowler for helpful discussions.

2.1 Introduction

To analyze calcium imaging data, scientists need to first detect neurons (also known as regions-of-interest (ROIs)) and extract temporal signals corresponding to each neuron from calcium imaging. These two goals together are called demixing. Calcium imaging records large neural populations, and the recorded neurons can overlap with each other, which makes the manual neuron identification infeasible. Hence, developing a reliable, automated demixing method is crucial.

There is an expansive literature focusing on this demixing problem. Most of these methods are based on matrix factorization. [11] proposed a PCA-ICA pipeline, which applied principle component analysis (PCA) to denoise the data first followed by independent component analysis (ICA) to extract signals. [12] developed a matrix factorization method based on a sparse dictionary learning [87]. [13] first introduced the nonnegative matrix factorization (NMF) method to solve the demixing problem. However, none of these methods can automatically determine the number of neurons or provide compact neuron shape estimates.

[14] added several key constraints to NMF, e.g. local constraints on neuron shapes, to better extract signals from calcium imaging, and the proposed method is known as the constrained NMF (CNMF) approach. The CNMF approach provides state-of-the-art denoising and demixing of calcium imaging, but it can leave significant visible signal behind in the residual (discarding potentially valuable signal) and is highly dependent on the initialization of the matrix factorization. Many efforts have been made towards improving the CNMF approach. For example, Suite2p [15] imposed different constraints and background models on the original CNMF approach. CNMF-E [19] extended the CNMF method to better handle the one-photon data with large background activities. [16] proposed a more robust method to noise and background contamination based on a one-sided Huber estimator. SCALPEL [18] presented a better initialization method for neuron shapes. Several other attempts have been made for demixing, such as the activity-based method in [17] and graph-based method in [21]. [88, 89] also developed online demixing methods for real-time analysis.

The NMF model used in the literature is very natural for calcium imaging applications, since each neuron has a shape that is fixed over the timescale of a typical imaging experiment (and these shapes can be represented as non-negative images, i.e., an element of the A matrix), and a corresponding time-varying calcium concentration that can be represented as a non-negative vector (an element of C): to form a movie we simply take a product of each of these terms and add them together with noise and background, i.e., form $Y = AC + B + E$.

However, current NMF-based approaches still leave room for improvement in several key directions. First, since NMF is a non-convex problem, good initializations are critical to obtain good results via the standard alternating optimization approaches (similar points are made in [18]). Good initialization approaches have been developed for somatic or nuclear calcium imaging, where simple Gaussian shape models are useful crude approximations to the elements of A [14], but these approaches do not apply to dendritic or axonal imaging. Second (related), it can be hard to separate weak components from noise using current NMF-based approaches. Finally, voltage imaging data does not neatly fit in the NMF framework, since

voltage traces typically display both positive and negative fluctuations around the baseline resting potential.

To improve the robustness of NMF approaches for demixing functional data, we make use of the growing literature on "guaranteed NMF" approaches — methods for computing a non-negative matrix factorization that are guaranteed to output the "correct" answer under suitable conditions and assumptions [90, 91, 92, 93]. In practice, these methods work well on clean data of sufficiently small dimensionality, but are not robust to noise and scale poorly to high-dimensional data. We can solve both of these issues by "superpixelizing" the denoised version of Y; the resulting NMF initialization method improves significantly on state-of-the-art methods for processing dendritic and axonal data.

The rest of this chapter is organized as follows. We first introduce a novel demixing method, which consists of the "superpixel" initialization and a subsequent new constrained NMF method. Next, we apply the proposed method to two challenging applications: dendritic calcium imaging data and voltage imaging data in the context of optogenetic stimulation. In both cases, we show that our new approach leads to faster and much more robust extraction of activity from the video data. Finally, we conclude this chapter with discussions and future works.

Open source code for the proposed method is available at https://github.com/paninski-lab/funimag.

2.2 Methods

We begin by defining notation. Our starting point is an imaging dataset that has been motion-corrected (i.e., we assume that there is no motion of visible cellular components from frame to frame of the movie) and then "unfolded" into a $d \times T$ matrix Y, where T is the number of frames in the movie and d is the number of pixels per frame (or voxels per frame if we are performing imaging in three dimensions). Now the typical approach is to model the data Y as $Y = AC + B + E$, where the columns of $A \in \mathbb{R}^{d \times K}$ model the locations of each

source (with K sources total), the rows of $C \in \mathbb{R}^{K \times T}$ model the time-varying fluorescence of each source, $B \in \mathbb{R}^{d \times T}$ is a "background" term to handle signals that can not easily be split into single-neuronal components, and $E \in \mathbb{R}^{d \times T}$ denotes temporally and spatially uncorrelated noise.

It is useful to break the processing pipeline into three sub-problems:

1. **Denoising**: separation of neural signal $Y^* = AC + B$ from noise E;
2. **Compression** of signal Y^*;
3. **Demixing**: factorization of Y^* into its constituent components A, C, and B.

Most prior work has attempted to solve these sub-problems simultaneously, e.g., to recover A and C directly from the raw data Y. As emphasized above, this direct approach involves a challenging non-convex optimization problem; the solution to this problem typically misses some structure in Y, is highly sensitive to initialization and hyperparameter settings, and can be particularly unstable in low-SNR regimes. We have found empirically that a sequential approach is more robust and effective. First we compute the compressed and denoised estimate $\hat{Y} = UV$; here U and V are chosen so that $\hat{Y}$ captures all of the signal in Y while retaining minimal noise (i.e., $\hat{Y} \approx Y^*$) and also U and V are highly-structured, compressible matrices, but we do not enforce any constraints between (U, V) and (A, C, B). The computation of U and V essentially solves sub-problems 1 and 2 simultaneously. The output matrices U and V are low-rank compared to Y, and U is additionally highly sparse (since U is formed by appending spatial components u from multiple local spatial patches, and each u_k is zero outside of its corresponding patch). Second, we exploit U, V, and the resulting denoised $\hat{Y}$ to facilitate the solution of problem 3.

The first two sub-problems are investigated in [20] using a novel Penalized Matrix Decomposition (PMD) approach. We directly apply the PMD method, get the denoised data $\hat{Y}$, and build an effective and scalable demixing algorithm for problem 3 on $\hat{Y}$. We elaborate our demixing algorithm 1 in detail below.

2.2.1 Initialization via pure superpixels

The first step of the initialization procedure is to identify groups of highly correlated spatially connected pixels – "superpixels." The idea is that a pixel within a neuron should be highly correlated with its neighbors, while a pixel containing mostly noise should have a much lower neighbor correlation. These neighbor correlations, in turn, can be estimated much more accurately from the denoised compared to the raw data. The superpixelization procedure results in a set of non-overlapping groups of pixels which are likely to be contained in good neural components. Then we want to extract "pure" superpixels, i.e., the subset of superpixels dominated by signal from just one neural component. We will use the temporal signals extracted from these pure superpixels to seed C in the NMF decomposition.

To identify superpixels, we begin with the denoised data $\hat{Y} = UV$. Since the compression process discussed in the previous section is rather conservative (aiming to preserve the full signal, at the expense of retaining a modest amount of noise), there is room to apply a more aggressive lossy denoiser in the initialization stage to further reduce any remaining noise in $\hat{Y}$. We soft-threshold signals in each pixel that are not sufficiently large — less than the median plus $\delta\times$ the median absolute deviation (MAD) within each pixel, with $\delta \approx 1$ or 2. (This thresholding serves to extract mostly spiking activity from functional imaging data.) We identify two neighboring pixels to be from the same superpixel if their resulting denoised, soft-thresholded temporal signals have a correlation larger than a threshold ϵ, with $\epsilon \approx 0.9$. Superpixels that contain fewer than τ pixels are discarded to further reduce noise and the total number of superpixels. We then apply rank 1 NMF on the signals from each superpixel to extract their (thresholded) temporal activities.

To extract pure superpixels, we apply the Successive Projection Algorithm (SPA) [94] to the temporal activities of superpixels. This algorithm removes "mixed" superpixels whose temporal activity can be modeled as a nonnegative linear combination of activity in other superpixels (up to some R-squared level larger than $1-\kappa$, where we use $\kappa \approx 0.2$) and outputs the remaining "pure" superpixels. See Algorithm 1 for pseudocode.

Note that running SPA on superpixels rather than raw pixels improves performance significantly here, since averaging signals within superpixels boosts SNR (making it easier to separate signal from noise and isolate pure from mixed pixels) and also greatly reduces the dimensionality of the non-negative regression problem SPA has to solve at each iteration. (To keep the problem size small we also run SPA just on small local spatial patches, as in the previous section.) Finally, while we have obtained good results with SPA, other approaches are available [95] and could be worth further exploration in the future. See Figure 2.1 for a visual summary of the full procedure.

2.2.2 Local NMF

Next we run NMF, using the temporal signals extracted from the "pure" superpixels to initialize C. Given the initial C, the typical next step is to regress onto the data to initialize A. (Note that pure superpixels typically capture just a subset of pixels within the corresponding neuron, so it is not efficient to initialize A with the pure superpixels.) However, given the large number of pixels in a typical functional imaging video, direct regression of C onto Y is slow and overfits, providing poor estimates of A.

This issue is well-understood [14], and several potential solutions have been proposed. For somatic imaging it makes sense to restrict the support of A to remain close to their initial values (we could use a dilation of the superpixel support for this). But for data with large dendritic or axonal components this approach would cut off large fractions of these components. Sparse regression updates are an option here, but these do not enforce spatial structure in the resulting A directly; this often results in "speckle" noise in the estimated spatial components (c.f. Figure 2.4 below).

We have found the following approach to be more effective. We initialize the support set Ω_k as the support of the k-th "pure" superpixel. Given C, we compute the correlation image for each component k as the correlation between the denoised data $\hat{Y}$ and the k-th temporal component, C_k. We truncate this correlation image below a certain threshold ϵ_1 to

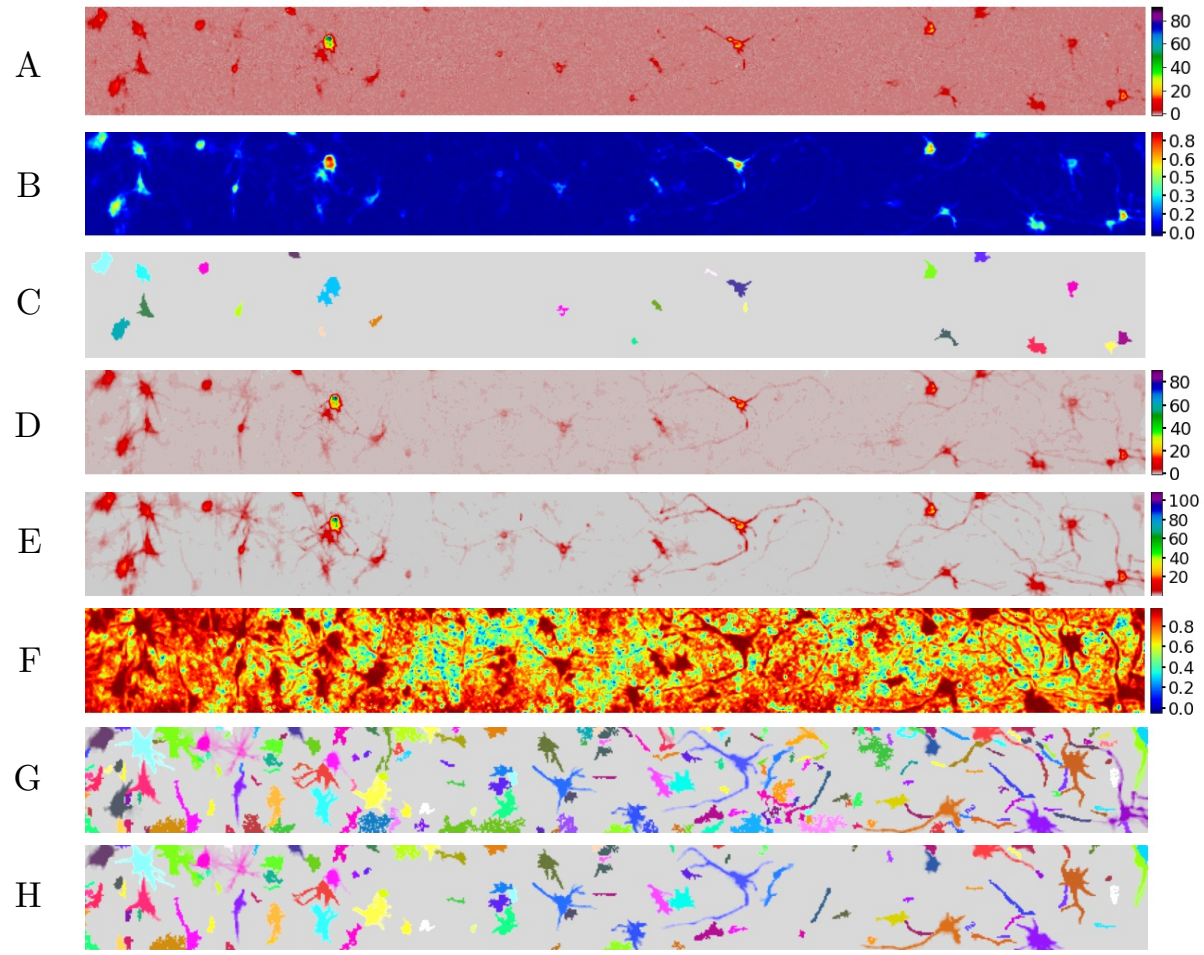

Figure 2.1: Denoising helps extract more complete superpixels in voltage imaging data (see Appendix for full dataset details). (A) Mean intensity projection of detrended data Y. (A spline detrender was applied to the raw data prior to analysis; see Appendix for details. This detrending should not be confused with an application of the trend filtering denoiser.) (B) Local correlation image of detrended data Y. (C) Superpixels extracted in detrended data Y with correlation cut-off $\epsilon = 0.2$, size cut-off $\tau = 10$. (D) Mean intensity projection of denoised data $\hat{Y}$. (E) Mean intensity projection of soft-thresholded denoised data. (F) Local correlation image of soft-thresholded denoised data; note that neural shapes are much clearer here than in panel A. (G) Superpixels extracted in soft-thresholded data with correlation cut-off $\epsilon = 0.95$, size cut-off $\tau = 15$. Note that we are using much more stringent criteria for defining superpixels here compared to panel C, but nonetheless (due to denoising) extract a much more complete superpixelization. (H) "Pure" superpixels extracted in soft-thresholded data with $\tau = 0.2$. See the superpixelization video for a time-varying illustration of these processing steps.

zero, then update Ω_k as the connected component of the truncated correlation image which overlaps spatially with the previous Ω_k. We use the modified fastHALS algorithm in [88] to

update A, C, and B to locally optimize the objective

$$\min_{A,C,\boldsymbol{b}} \|\hat{Y} - AC - B\|_F^2, \text{ s.t. } A_k^x = 0 \; \forall x \notin \Omega_k, A \geqslant 0, C \geqslant 0, B = \boldsymbol{b}1^T, \boldsymbol{b} \geqslant 0. \tag{2.1}$$

Here we have modeled the background B as a simple temporally-constant vector; we discuss generalizations to time-varying backgrounds below. Also note that we are approximating $\hat{Y}$ directly here, not the thresholded version we used to extract the superpixels above.

Finally, we incorporate a merge step: we truncate the correlation image below certain threshold ϵ_2 to zero, and automatically merge neurons if their truncated correlation images are highly overlapped. The full algorithm is shown in Algorithm 2.

2.2.3 Further implementation details

Multi pass strategy: As in [19], we find it effective to take a couple passes over the data; particularly in datasets with high neuron density, the first NMF pass might miss some dim neurons. We decrease the MAD threshold δ and re-run Algorithm 1 on the residual to find additional components, and then run a final merge and NMF update to complete the pipeline.

Improvements from denoising and compression: Compressed data leads to faster NMF updates, since we can replace $\hat{Y}$ as UV; in fastHALS, we can regress each $\boldsymbol{a}_k$ on U or $\boldsymbol{c}_k$ on V first instead of directly onto Y. Similarly, when calculating the correlation image, we can compute the correlation between the low rank V and $\boldsymbol{c}_k$ first. As emphasized above, denoising also improves the estimation of the correlation images, which in turn improves the estimation of the support sets Ω_k.

Time-varying background: It is straightforward to generalize the objective 2.1 to include a time-varying background, using either a low-rank model (as in [14]) or a ring-structured model (as in [19]). For the low-rank background model, we have found that performing an SVD on the data excluding the support of the superpixels provides an efficient initialization

for the background temporal components.

Handling 3D imaging data: We also have 3D movie dataset, in which calcium signals at different depths of the brain are recorded simultaneously. Our initialization method can be easily adapted to extracting 3D supervoxels.

Incorporating temporal penalties: Note that we are only imposing nonnegativity in C here; after denoising to obtain $\hat{Y}$, we have found that this simple nonnegative constraint is sufficient for the datasets examined here. However, it is certainly possible to incorporate temporal penalties or constraints on C (e.g., a TF penalty or a non-negative auto-regressive penalty as in [14]), either within each iteration or as a final denoising step.

Post-processing: We find that sorting the extracted components by their "brightness," computed as $\max \boldsymbol{a}_k \cdot \max \boldsymbol{c}_k$, serves to separate dim background components from bright single-neuronal components. We also found it useful to drop components whose temporal trace has skewness less than 0.5; traces with high skewness correspond to components with significant spiking activity, but low-skewness traces corresponded to noise.

2.3 Results

We have applied the denoising and compression approach described above to a wide variety of functional imaging datasets (See [20] for full details):

- **Dendritic**: two-photon Bessel-beam calcium imaging of dendrites in somatosensory cortex of mouse in vivo
- **Voltage**: one-photon in vitro voltage imaging under optogenetic stimulation.

2.3.1 Voltage imaging data

We begin with an analysis of a challenging voltage imaging dataset. Voltage imaging (VI) data presents a few important challenges compared to calcium imaging (CI) data: currently-available VI data typically has much lower SNR and displays much stronger bleaching effects

Algorithm 1 Pseudocode for the complete proposed pipeline.

Input: Motion corrected data $Y \in \mathbb{R}^{d \times T}$, MAD threshold δ, minimum size of superpixels τ, correlation threshold for superpixels ϵ, R^2 threshold in SPA κ.

$\sigma(\boldsymbol{x}) \leftarrow$ estimated noise for each pixel $\boldsymbol{x}$ of Y;
$\mu(\boldsymbol{x}) \leftarrow$ mean for each pixel of Y;
$Y \leftarrow (Y - \mu(\boldsymbol{x}))/\sigma(\boldsymbol{x})$;
$(\hat{Y}, U, V) \leftarrow \mathrm{PMD}(Y)$;
$n \leftarrow 0$; $A \leftarrow [\,]$, $C \leftarrow [\,]$, $\boldsymbol{b} \leftarrow$ median for each pixel of $\hat{Y}$;
while $n <$ maximum number of passes **do**
    $R \leftarrow \hat{Y} - AC - \boldsymbol{b}$;
    $\sigma_{med}(\boldsymbol{x}) \leftarrow$ median absolute deviation for each pixel of R;
    $\mu_{med}(\boldsymbol{x}) \leftarrow$ median for each pixel of R;
    $\tilde{Y} \leftarrow \max\left(0, R - \mu_{med}(\boldsymbol{x}) - \delta \cdot \sigma_{med}(\boldsymbol{x})\right)$;
    $\mathrm{corr}(\boldsymbol{x}, \boldsymbol{x}^*) \leftarrow \mathrm{corr}\left(\tilde{Y}(\boldsymbol{x}, t), \tilde{Y}(\boldsymbol{x}^*, t)\right)$ for all neighbouring pixel pairs $(\boldsymbol{x}, \boldsymbol{x}^*)$;
    Extract superpixels: connect $\boldsymbol{x}$ and $\boldsymbol{x}^*$ together if $\mathrm{corr}(\boldsymbol{x}, \boldsymbol{x}^*) \geqslant \epsilon$ to construct connected components and discard those smaller than τ, forming superpixels $\Omega_k, k = 1, \cdots, K$;
    $(\boldsymbol{a}_k, \boldsymbol{c}_k) \leftarrow$ rank 1 NMF of $\tilde{Y}$ on support $\Omega_k, k = 1, \cdots, K$;
    $[i_1, i_2, \cdots, i_S] \leftarrow \mathrm{SPA}([\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_K], \kappa)$; $i_1, i_2, \cdots, i_S$ are indices of pure superpixels;
    $A_0 \leftarrow [A, \boldsymbol{a}_{i_1}, \boldsymbol{a}_{i_2}, \cdots, \boldsymbol{a}_{i_S}]$;
    $C_0 \leftarrow [C^T, \boldsymbol{c}_{i_1}, \boldsymbol{c}_{i_2}, \cdots, \boldsymbol{c}_{i_S}]^T$;
    $\boldsymbol{b}_0 \leftarrow \boldsymbol{b}$;
    $(A, C, \boldsymbol{b}) \leftarrow \mathrm{LocalNMF}(U, V, A_0, C_0, \boldsymbol{b}_0)$;
    $\delta \leftarrow \delta - 1$;
    $n \leftarrow n + 1$;
end while
$\eta(k) \leftarrow$ estimated noise for $\boldsymbol{c}_k$ using average of high frequency domain of PSD;
(Optional) Denoise temporal components, e.g. by ℓ_1 trend filter: $\boldsymbol{c}_k \leftarrow \min_{\tilde{\boldsymbol{c}}_k} \|\tilde{\boldsymbol{c}}_k\|_1$, s.t. $\|\tilde{\boldsymbol{c}}_k - \boldsymbol{c}_k\|_F \leqslant \eta(k)\sqrt{T}, k = 1, \cdots, K$;
Output: $A, C, \boldsymbol{b}$

than CI data. The dataset we focus on here has another challenging feature: the preparation was driven with time-varying full-field optogenetic stimulation, resulting in highly correlated subthreshold activity in the visible cells, which are highly overlapping spatially. In preliminary analyses of this data we applied variants of CNMF-E [19] but did not obtain good results (data not shown), due to the strong bleaching and optogenetic stimulation-induced correlations present in this data.

Thus we pre-processed this data by applying a spline-based detrending to each pixel (see

Algorithm 2 Pseudocode for LocalNMF.

Input: Compressed factors $U \in \mathbb{R}^{d\times r}, V \in \mathbb{R}^{T\times r}$ ($r = rank(\hat{Y})$); initial constant background $\boldsymbol{b}_0$, spatial components $A_0 = [\boldsymbol{a}_{1,0}, \cdots, \boldsymbol{a}_{K,0}] \in \mathbb{R}^{d\times K}$, and temporal components $C_0 = [\boldsymbol{c}_{1,0}, \cdots, \boldsymbol{c}_{K,0}]^T \in \mathbb{R}^{K\times T}$; truncation threshold when updating support ϵ_1, truncation threshold when merging ϵ_2, overlap threshold when merging ϵ_3.

$\Omega_k \leftarrow \text{supp}(\boldsymbol{a}_{k,0})$ is spatial support for k-th component, $k = 1, \cdots, K$;
$\hat{A} \leftarrow A_0, \hat{C} \leftarrow C_0, \hat{\boldsymbol{b}} \leftarrow \boldsymbol{b}_0$;
$\nu(\boldsymbol{x}) \leftarrow$ standard deviation for each pixel of $\hat{Y} = UV$;
$\bar{V} \leftarrow$ mean for each column of V;
while not converged **do**
    $P \leftarrow [U, -\boldsymbol{b}] \left(\begin{bmatrix} V \\ 1^T \end{bmatrix} \hat{C}^T \right)$;
    $Q \leftarrow \hat{C}\hat{C}^T$;
    for $k = 1 : K$ **do**
        Update spatial: $\hat{\boldsymbol{a}}_k(\Omega_k) \leftarrow \max\left(0, \hat{\boldsymbol{a}}_k(\Omega_k) + \frac{P(\Omega_k,k) - \hat{A}(\Omega_k)Q(:,k)}{Q(k,k)}\right)$;
    end for
    Update constant background: $\hat{\boldsymbol{b}} \leftarrow \max\left(0, \frac{1}{T}(UV - \hat{A}\hat{C})1\right)$;
    $P \leftarrow \left[V^T, 1\right]\left([U, -\boldsymbol{b}]^T \hat{A}\right)$;
    $Q \leftarrow \hat{A}^T\hat{A}$;
    for $k = 1 : K$ **do**
        Update temporal: $\hat{\boldsymbol{c}}_k \leftarrow \max\left(0, \hat{\boldsymbol{c}}_k + \frac{P(:,k) - \hat{C}Q(:,k)}{Q(k,k)}\right)$;
    end for
    for every 4 iterations **do**
        for $k = 1 : K$ **do**
            $\text{corr}(k, \boldsymbol{x}) \leftarrow \frac{1}{T\cdot\nu(\boldsymbol{x})\cdot \text{sd}(\boldsymbol{c}_k)} U(\boldsymbol{x}, :)\left((V - \bar{V})(\boldsymbol{c}_k - \bar{\boldsymbol{c}}_k)\right)$;
            Update spatial support: $\Omega_k \leftarrow$ biggest connected component in $\{\boldsymbol{x} | \text{corr}(k, \boldsymbol{x}) \geqslant \epsilon_1\}$ that spatially overlaps with $\{\boldsymbol{a}_k > 0\}$;
            $\hat{\boldsymbol{a}}_k(\Omega_k^c) \leftarrow 0$;
            $\rho(k, \boldsymbol{x}) \leftarrow (\text{corr}(k, \boldsymbol{x}) \geqslant \epsilon_2)$;
        end for
        Merge overlapping components k_1, k_2 if $\sum_{\boldsymbol{x}} (\rho(k_1, \boldsymbol{x}) * \rho(k_2, \boldsymbol{x})) / \sum_{\boldsymbol{x}} \rho(k_i, \boldsymbol{x}) \geqslant \epsilon_3$;
        $(\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{c}}) \leftarrow$ rank-1 NMF on $[\hat{\boldsymbol{a}}_{k_1}, \cdots, \hat{\boldsymbol{a}}_{k_r}][\hat{\boldsymbol{c}}_{k_1}, \cdots, \hat{\boldsymbol{c}}_{k_r}]$ for merged components $k_1, \cdots, k_r$;
        $\hat{A} \leftarrow \left[\hat{A} \backslash \{\boldsymbol{a}_{k_1}, \cdots, \boldsymbol{a}_{k_r}\}, \tilde{\boldsymbol{a}}\right], \hat{C} \leftarrow \left[\hat{C}^T \backslash \{\boldsymbol{c}_{k_1}, \cdots, \boldsymbol{c}_{k_r}\}, \tilde{\boldsymbol{c}}\right]^T$;
        update number of components K;
    end for
end while
Output: $\hat{A}, \hat{C}, \hat{\boldsymbol{b}}$

[20] for full details). This served to attenuate the highly-correlated bleaching signals and subthreshold fluctuations in the raw data, leaving behind spiking signals (which were not per-

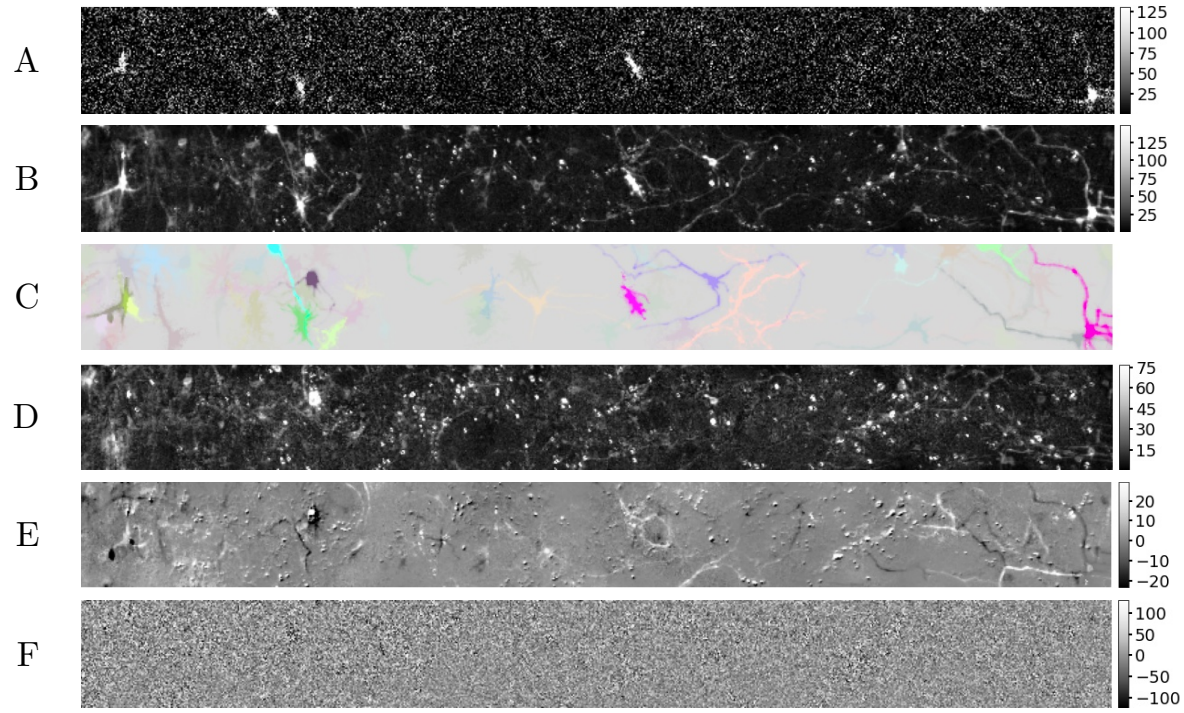

Figure 2.2: An example frame illustrating demixing on voltage imaging data. (A) Detrended data Y. (B) Denoised data $\hat{Y}$. (C) Extracted signals AC; each component k is assigned a unique color, and the intensity of each pixel at each time is determined by the corresponding value of AC. (D) Constant background $\boldsymbol{b}$. (E) Residual $\hat{Y} - AC - \boldsymbol{b}1^T$. (F) Noise removed in the denoising step. See the voltage imaging demixing video for a time-varying representation of the results here.

fectly correlated at the millisecond resolution of the video data here) along with uncorrelated noise as the dominant visible signals in the data. Figure 2.1 shows that the denoiser (followed by soft-thresholding) serves to significantly improve the separability of neural signals from noise in this data: the superpixels obtained after denoising and soft-thresholding provide excellent seeds for the constrained NMF analysis. Figures 2.2 (and the corresponding video) and 2.3 demonstrate that the full demixing pipeline achieves good performance, extracting components with high spatial and temporal SNR and leaving relatively little residual signal behind despite the limited SNR and the multiple overlapping signals visible in the original (detrended) data. Note that in the final step we project the estimated spatial components back onto the original data, recovering the (highly correlated) temporal components includ-

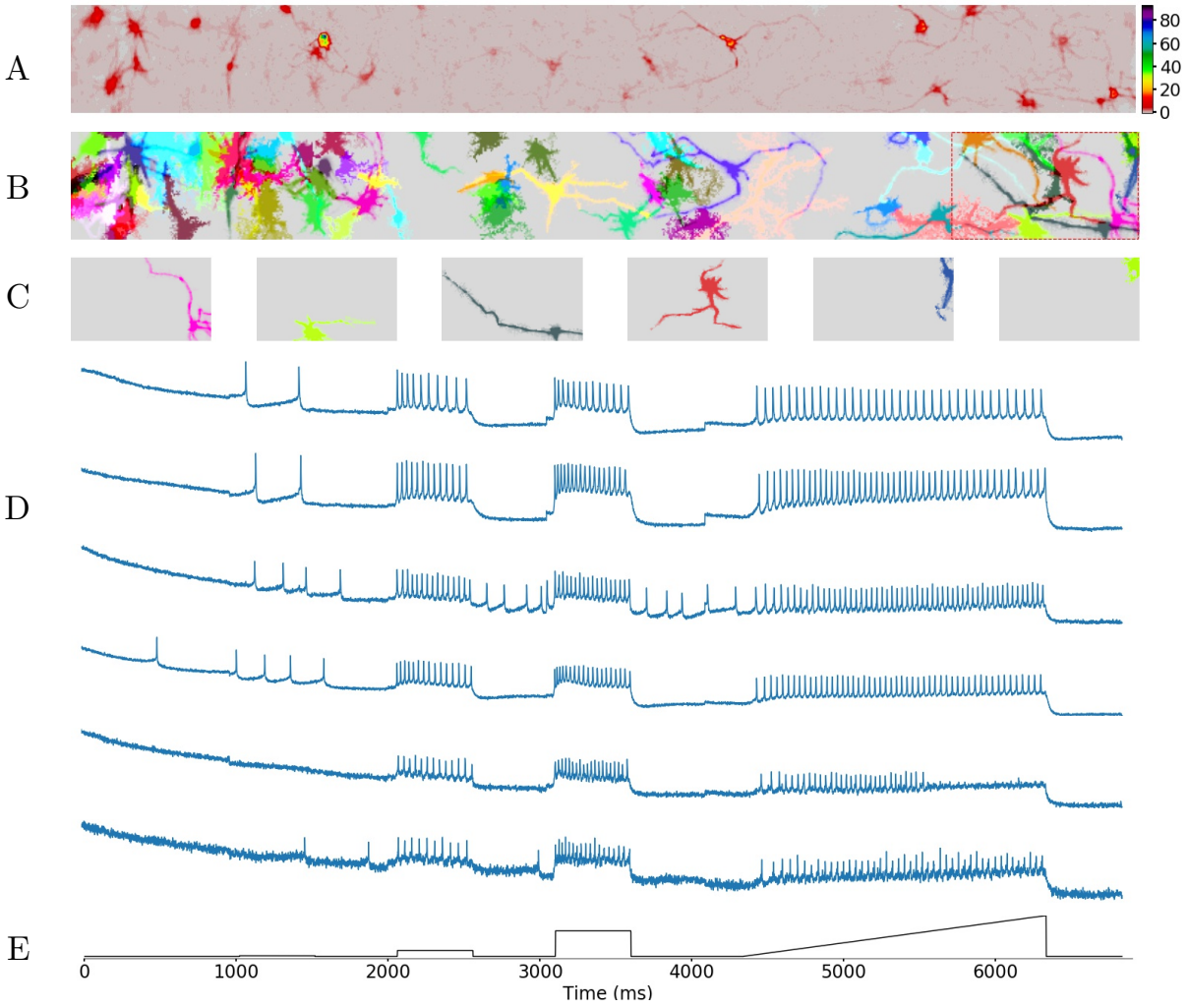

Figure 2.3: Components extracted from voltage imaging data. (A) Mean intensity projection of $\hat{Y}$. (B) Extracted spatial components (each assigned a unique color). (C) Details of the spatial components extracted in the zoomed-in patch (red outline in panel B), sorted in decreasing order of brightness. (D) Raw temporal components corresponding to the spatial components shown in C (blue lines). Note that the highly-correlated subthreshold activity and the strong bleaching trends visible in these components. (E) Optogenetic stimulation (consisting of three steps of increasing amplitude followed by a ramp; black line).

ing strong bleaching components (panel D of Figure 2.3). Finally, we achieved a speedup in the NMF iterations here that was roughly proportional to the ratio of the rank of Y compared to the rank of U. A variant of the proposed method has also been applied to extract signals from the 1-photon voltage imaging in vivo [96].

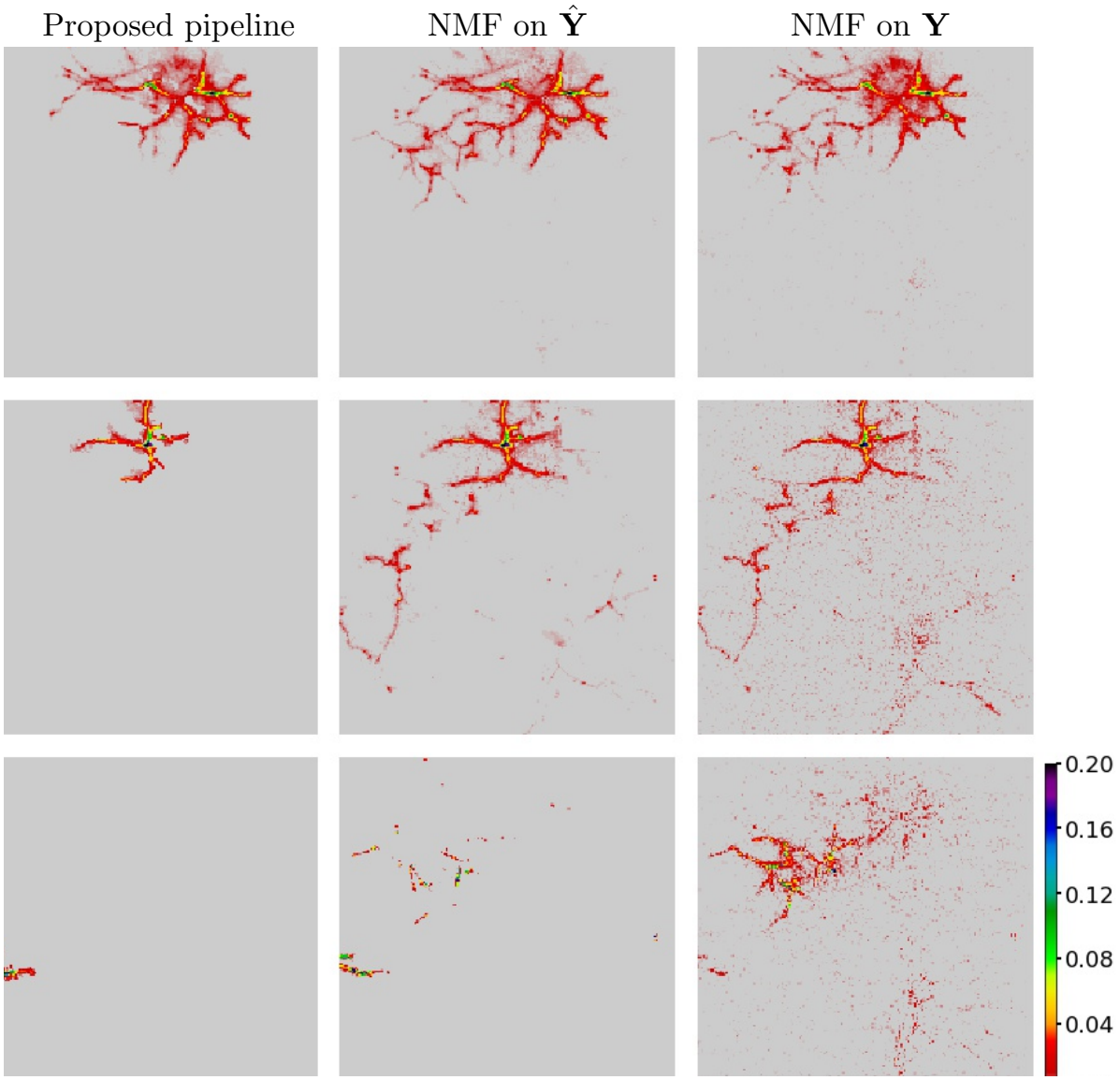

Figure 2.4: Comparison of spatial components extracted from Bessel dendritic imaging data. Each row shows best-matching components extracted by our proposed method (first column), sparse NMF on denoised data $\hat{Y}$ (second column) and sparse NMF on raw data Y (third column). See the Bessel dendritic imaging demixing video for further details. The proposed pipeline extracts components that are significantly more localized and less noisy than the components extracted by sparse NMF; also note that denoising helps sparse NMF extract cleaner spatial components.

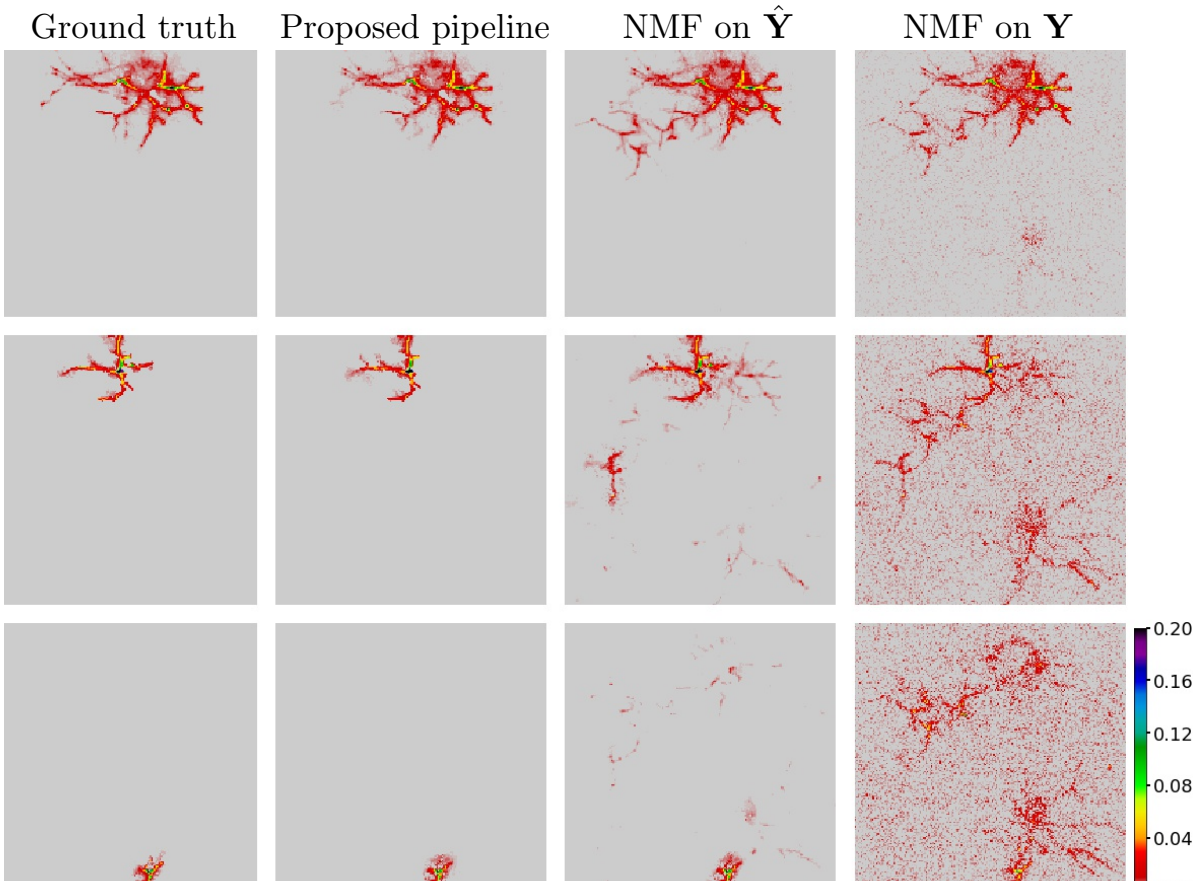

Figure 2.5: Comparison to simulated ground truth based on Bessel dendritic imaging data. Spatial components are arranged as in the previous figure, with the addition of ground truth components shown in the first column. Note that the proposed pipeline recovers the ground truth simulated components much more accurately than do the sparse NMF baseline approaches.

2.3.2 Bessel dendritic imaging data

The VI dataset analyzed in the preceding subsection contained a number of large visible axonal and dendritic components, but also displayed strong somatic components. For our next example we focus on a CI dataset dominated by dendritic components, where the simple Gaussian spatial filter approach introduced in [14] for initializing somatic components is ineffective. (Indeed, in dendritic or axonal imaging datasets, a search for "hotspots" in the images is biased towards pixels summing activity from multiple neurons — and these "non-pure" pixels are exactly those we wish to avoid in the demixing initialization strategy

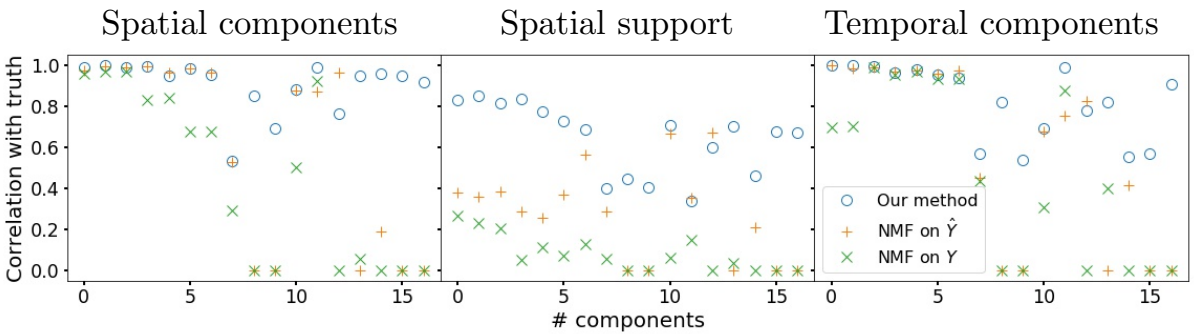

Figure 2.6: Quantification of comparisons on simulated Bessel dendritic imaging data. Components are ordered by brightness; top 17 brightest components shown here. First column shows the correlation between true vs spatial components estimated by proposed pipeline (o), sparse NMF on $\hat{Y}$ (+), and sparse NMF on Y (x). Second column shows the correlation between the supports of the true and estimated spatial components. Third column shows the correlation between the true vs estimated temporal components. (The baseline NMF approaches missed some dimmer, weaker neurons, so the corresponding symbols are set to zero here.) Note that components extracted by proposed pipeline typically have higher correlation with true components than sparse NMF baseline approaches.

proposed here.)

Figure 2.4 illustrates several of the spatial components extracted by our pipeline (again, see the corresponding video for a more detailed illustration of the demixing performance); these components visually appear to be dendritic segments and match well with the signals visible in the data movie. Notably, no parameter tuning was necessary to obtain good demixing performance on both the VI and CI datasets, despite the many differences between these data types. Additionally, as a baseline comparison we applied a simple sparse NMF approach with random initialization (similar to the method described in [14]) to both the denoised and raw data ($\hat{Y}$ and Y, respectively). As shown in the right columns of Figure 2.4, this baseline approach extracted components that were much more mixed and noisy than the components extracted by our proposed demixing pipeline; we also found that the baseline approach was more prone to missing weaker, dimmer components than was the proposed pipeline (data not shown).

The above analyses depended on qualitative visual examinations of the obtained com-

ponents and demixing video. We also generated simulated data with characteristics closely matched to the raw data, in order to more quantitatively test the demixing performance against a known (albeit simulated) ground truth. To generate simulated data Y, we used the A and C estimated from the raw data, and further estimated the conditional distribution of the residual as a function of the denoised data AC in the corresponding pixel x and time bin t; then we added independent noise samples from this signal-dependent conditional distribution (but with the noise scale multiplied 2x, to make the simulation more challenging) to AC. See the simulated Bessel dendritic imaging video for comparison of real and simulated data. We ran the three demixing pipelines on this simulated data. Typical results of these simulations are shown in Figure 2.5: again we see that the proposed pipeline captures the ground truth components much more accurately than do the baseline methods, similar to the results shown in Figure 2.4. Quantitatively, components extracted by proposed pipeline have higher correlation with ground truth components than do those extracted by sparse NMF approaches, as shown in Figure 2.6.

2.4 Discussion

We have presented new scalable approaches for demixing functional imaging data. The new demixing methods proposed here are particularly useful for data with many dendritic and axonal processes, where methods based on simple sparse NMF are less effective.

2.4.1 Related work

There are also some interesting connections between the demixing approach proposed in [18] and our approach to initializing NMF, which is based on the sparse projection algorithm (SPA). [97, 95] discuss the relationships between SPA and group-sparse dictionary selection methods related to the approach used in [18]; thus the methods we use to compute "pure" superpixels and the methods used in [18] to select neural dictionary elements are closely related. However, our denoise-then-superpixelize approach to seeding the dictionary of neural

temporal components is in a sense converse to the clustering approach developed in [18] for seeding the dictionary of neural spatial components. There may be room to fruitfully combine these two approaches in the future.

2.4.2 Future work

In the near future we plan to incorporate our code into the CaImAn and CNMF-E packages for calcium imaging analysis.

We can continue to develop a real-time online version of our demixing algorithm as in [89]. We may also explore alternative methods (e.g. using artificial neural network) for improving identifying superpixels in initialization and constructing spatial support in localNMF.

2.5 Appendix

2.5.1 Video captions

1. Superpixelization video

   Panels from top to bottom: (1) detrended movie $\mathbf{Y}$; (2) denoised movie $\hat{\mathbf{Y}}$; (3) MAD soft-thresholded movie; (4) rank-1 NMF approximation within superpixels; (5) superpixels; (6) pure superpixels.

2. Voltage imaging demixing video

   Panels from top to bottom: (1) detrended movie $\mathbf{Y}$; (2) denoised movie $\hat{\mathbf{Y}}$; (3) estimated signal $\mathbf{AC}$; (4) background $\mathbf{B}$; (5) residual $\hat{\mathbf{Y}} - \mathbf{AC} - \mathbf{B}$; (6) estimated noise $\mathbf{Y} - \hat{\mathbf{Y}}$.

3. Bessel dendritic imaging demixing video

   Top: (left) motion corrected movie $\mathbf{Y}$; (middle) denoised movie $\hat{\mathbf{Y}}$; (right) estimated signal $\mathbf{AC}$; Bottom: (left) background $\mathbf{B}$, (middle) residual $\hat{\mathbf{Y}} - \mathbf{AC} - \mathbf{B}$, and (right) estimated noise $\mathbf{Y} - \hat{\mathbf{Y}}$.

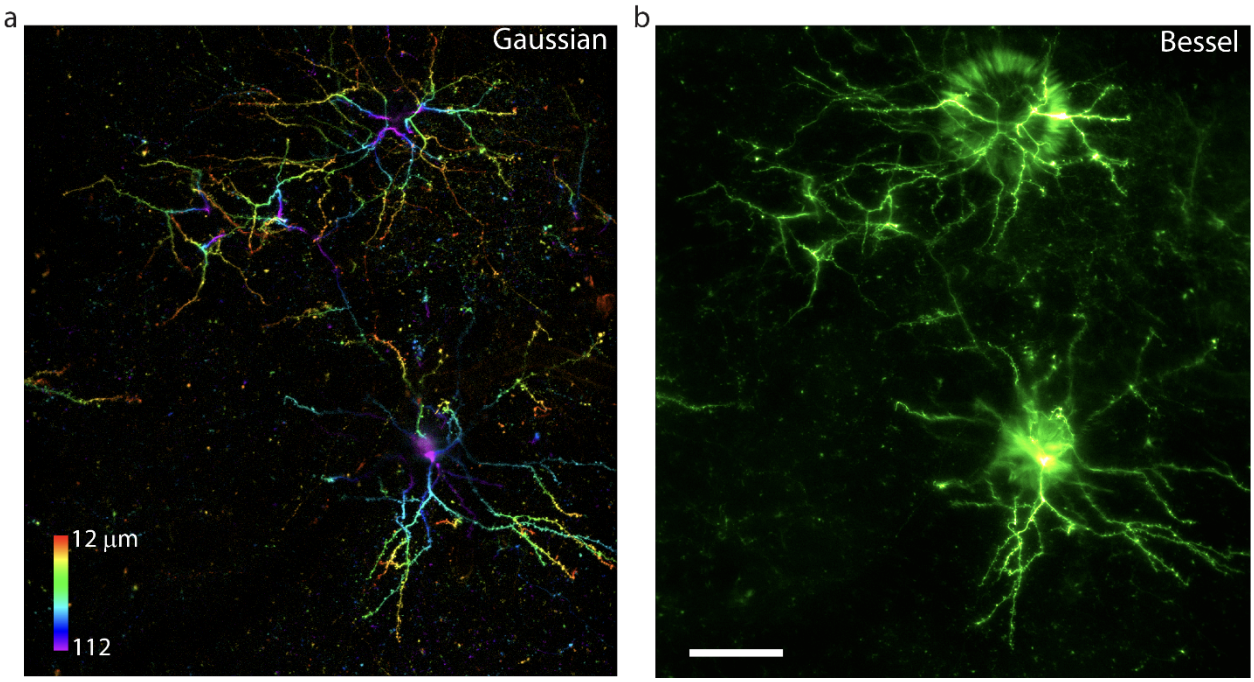

Figure 2.7: In vivo volumetric imaging of dendrites in the mouse brain. (a) Maximum intensity projection of a 3D volume (635 μm $\times$ 694 μm $\times$ 100 μm) of dendrites. The sampling size was 0.33 μm/pixel. Post-objective power: 24 mW. (b) Image of the same volume collected by scanning a Bessel focus with 0.60 μm lateral FWHM and 71 μm axial FWHM. The effective volume rate was 3.7 Hz. Post-objective power: 120 mW. Excitation wavelength: 970 nm. Scale bar: 100 μm.

4. Simulated Bessel dendritic imaging video

   Top: (left) Motion corrected real movie; (right) simulated movie. Bottom: (left) estimated noise from real movie; (right) simulated noise.

2.5.2 Datasets details

Bessel dendritic imaging data All surgical procedures were in accordance with protocols approved by the Howard Hughes Medical Institute Janelia Research Campus Institutional Animal Care and Use Committee.

C57BL/6J mice over 8 weeks old at the time of surgery were anesthetized with isoflurane anesthesia (1–2%). A craniotomy over nearly the entire left dorsal cortex (from Bregma +3 mm to Bregma −4.0 mm) was performed with the dura left intact, with the procedure

described in detail previously in [98]. AAV2/9-synapsin-flex-GCaMP6s (2.5×10^{13} GC/ml) was mixed with AAV2/1-synapsin-Cre (1.5×10^{13} GC/ml, 1000×dilution with PBS) at 1:1 to make the working viral solution for intracerebral injections. 30 nl viral solution was slowly injected into exposed cortex at 0.5 mm below dura. Injection sites were evenly spaced (at 0.7-0.9 mm separation) along two lines at 2.3 mm and 3.3 mm parallel to the midline. A custom-made glass coverslip (450 μm thick) was embedded in the craniotomy and sealed in place with dental acrylic. A titanium head bar was attached to the skull surrounding the coverslip. After recovery from surgery, the mice were habituated to head fixation. Four weeks after surgery, the head-fixed mouse was placed on a floating ball in the dark. The spontaneous neural activity as indicated by GCaMP6s fluorescence signal was recorded in the somatosensory cortex.

Volumetric imaging of dendrites was achieved by scanning an axially extended Bessel focus in [99] and [100]. An axicon-based Bessel beam module was incorporated into a 2-photon random access mesoscope (2p-RAM) in [99]. Details of the 2p-RAM have been described previously in [98]. Briefly, the system was equipped with a 12kHz resonant scanner (24 kHz line rate) and a remote focusing unit that enabled fast axial movements of the focal plane. The system has an excitation numerical aperture (NA) of 0.6 and a collection NA of 1.0. The measured lateral full width at half maximum (FWHM) of the Gaussian focus at the center of the field of view was 0.65 μm. The lateral and axial FWHMs of the Bessel focus were 0.60 μm and 71 μm, respectively. Scanning the Bessel focus in two dimensions, therefore, probed brain volumes within a 100 μm axial range. The volumetric dendritic data presented in this paper were obtained by placing the center of the Bessel focus at 62 μm below dura to probe structures at 12 μm to 112 μm below dura (Figure 2.7). Dendrites within this volume were imaged at an effective volume rate of 3.7 Hz, with each image having 1924×2104 pixels at 0.33 μm/pixel in the x-y plane. The wavelength of the excitation light was 970 nm and the post-objective excitation power was 120 mW. Images were spatially decimated and cropped for the analyses shown here.

Voltage imaging data Q-State's proprietary Optopatch all optical electrophysiology platform was used to record fluorescence recordings from induced pluripotent stem (iPS) cell-derived NGN2 excitatory neurons from a cohort of human subjects [101]. Stimulation of action potentials was achieved with a blue light-activated channelrhodopsin (CheRiff). Fluorescent readout of voltage was enabled by an Archaerhodopsin variant (QuasAr). NGN2 neurons were produced at Q-State using a transcriptional programming approach. Recordings were performed with an ultra-widefield instrument with a resolution of 800×80 pixels (corresponding field of view of 2 mm^2) at a frame rate of 987 Hz.

The obtained data displayed breaks during stimulus resets and photobleaching. To remove these effects from the raw data, we removed frames during stimulus resets, extracted slow trends with a robust B-spline regression (with knots chosen to allow for non-differentiability at stimulus change-points and discontinuity at stimulus resets), and then a quadratic regression against frames with no stimuli to capture and then remove photobleaching effects.

Chapter 3: A zero-inflated gamma model for deconvolved calcium imaging traces

This chapter was published as "A zero-inflated gamma model for deconvolved calcium imaging traces" [84] in Neurons, Behavior, Data Analysis and Theory (NBDT) 2020 with Xue-Xin Wei, Andres Grosmark, Zaki Ajabi, Fraser Sparks, Pengcheng Zhou, Mark Brandon, Attila Losonczy, and Liam Paninski. We thank Tian Zheng and John Cunningham for helpful discussions.

3.1 Introduction

Calcium imaging is one of the primary methods for measuring the activities of large neural populations at single-cellular resolution [1, 2, 3, 4, 102]. Calcium imaging has several important advantages: it offers high spatial resolution and can be coupled with various genetic tools to achieve cell-type specificity; it has proven to be scalable and can monitor hundreds/thousands of neurons in vivo simultaneously; finally, it allows for longitudinal tracking of cellular activity across multiple days or weeks [103, 104, 105].

At the same time, calcium imaging presents some important analysis challenges: calcium signals represent a slow, nonlinear encoding of the underlying spike train signals of interest, and therefore it is necessary to denoise and temporally deconvolve temporal traces extracted from calcium video data (e.g., Δ F/F) into estimates of neural activity. These issues have received extensive attention in the literature [22, 23, 14, 24, 25, 26, 27, 28, 29, 30, 31, 32]. Some of these deconvolution approaches estimate spiking probabilities directly [22, 14, 24, 28, 29, 32], but many approaches instead estimate the influx of calcium in each time bin, rather than a spiking probability [23, 14, 26, 27, 30, 31, 106]; these non-probabilistic approaches

tend to be faster and are therefore popular in practice.

What is a proper statistical model for the output of these non-probabilistic calcium deconvolution approaches? Defining such a model is the first step in any likelihood-based downstream analyses, e.g., Bayesian decoding, probabilistic latent factor modeling, and/or estimation of neural encoding models [5] — but somewhat surprisingly, no "standard model" has emerged yet for the deconvolved output.

The simplest approach is to simply threshold the output and treat the resulting super-threshold events as "spikes" (corresponding to a Bernoulli statistical model for these spikes), but this approach clearly discards information about the number of spikes per bin, and there is no obvious optimal way to set the threshold. Another naive approach would be to apply standard point-process models (e.g., Poisson regression models) to the deconvolved output — but as we will see below, the Poisson model is a poor approximation here, not least because the deconvolved output can take continuous values, while the Poisson distribution is supported on the integers.

In this chapter, we investigate statistical models to characterize the deconvolved calcium activity. (To be clear, we do not propose any new deconvolution approaches here; instead, we restrict our attention to modeling the output of existing deconvolution methods.) In particular, we propose a zero-inflated gamma (ZIG) model, a two-component mixture model including a "spike" of probability at zero response and another continuous component for modeling positive responses, specified by a gamma distribution. We apply this model to simulated data and real imaging datasets from hippocampus and thalamus, and find that it provides good fits across a wide variety of deconvolution parameters and data types. Next we show that the ZIG model can be embedded within "encoding models" to characterize the probability of calcium responses given time-varying covariates such as the location or orientation of the animal during behavior. Finally, we demonstrate that the ZIG-based encoding model leads to more accurate Bayesian decoding of these covariates.

3.2 Results

3.2.1 Nonnegative deconvolution methods applied to calcium fluorescence traces produce a mixture of zeros and positive real-valued output, well-captured by the zero-inflated gamma model

We begin with simple simulated data (Figure 3.1a-d): we generate a Poisson spike train, then push this spike train through a standard auto-regressive AR(1) model for calcium response [23] and add noise to generate simulated fluorescence traces, and then run a popular non-negative deconvolution method [22, 26] to obtain the post-deconvolution response, denoted as $\hat{s}_t$. With experimentally relevant signal-to-noise levels, the resulting histogram of deconvolved responses $\hat{s}_t$ typically has a "spike-and-slab" form (Figure 3.1b-d, right): significant mass is placed exactly at zero (the "spike"), with the remaining mass forming a continuous "slab" on the positive real axis. (This spike-and-slab structure of $\hat{s}_t$ is unsurprising: the deconvolution approach applied here enforces sparsity and non-negativity constraints on $\hat{s}_t$, forcing $\hat{s}_t$ to be exactly zero for many timesteps t.) Empirically, a shifted gamma model suffices to capture the shape of the slab (green traces in Figure 3.1b-d, right); the shift is fixed to be equal to the minimum spike size allowed by the deconvolution algorithm ("s_{min}"), and therefore the gamma distribution is still specified by two parameters. We denote the resulting three-parameter distribution (with the third parameter corresponding to the probability of a non-zero response) as the "zero-inflated gamma" (ZIG) model. As we will see below, it is critical to use at least a two-parameter distributional family for the slab, to capture changes in the mean and variance. The gamma family is a convenient two-parameter family that provides a good fit to the data, but other distributional families beyond the gamma could also be suitable here.

In these simulations, we have made several simplifying assumptions, including: i) a simple AR(1) model for the generative process of the calcium fluorescence; ii) the a priori knowledge about the time constant of the AR process; iii) the increase of calcium concentration

following each spike has a constant, deterministic size. Presumably, any deviations from these assumptions would make the estimated "spikes" $\hat{s}_t$ noisier, thus making the continuous part of the response histogram smoother. On the other hand, increasing the signal-to-noise ratio (SNR) and/or making the neurons burstier can introduce multiple "bumps" in the continuous part of the distribution (not shown). This multiple-bump case could potentially be handled by incorporating a multiple-component mixture model for the slab in our spike-and-slab model, but (as we will discuss next), in practice for real data we have not found this to be necessary and have not pursued this direction systematically.

We turn next to real data. In most real datasets, the ground truth spiking (the first part of our simulation pipeline outlined in Figure 3.1a) is not available, but nonetheless we can run the same deconvolution algorithm on the observed fluorescence trace to obtain $\hat{s}_t$. The resulting histogram of $\hat{s}_t$ is again well-fit by the ZIG model, for two example neuron cases shown in Figure 3.1e-f.

3.2.2 The ZIG model is applicable to the outputs of multiple deconvolution methods, applied to data from multiple calcium indicators

We next seek to determine whether the observations made in Figure 3.1 are specific to a particular deconvolution method or calcium indicator.

In Figure 3.2, we check three deconvolution methods, including an L_1-penalized method with a soft threshold [23, 14], a method with a hard threshold (i.e., positive minimal spike size "s_{min}") [26], and an L_0-penalized method [107, 27]. Each method has a free parameter which controls the sparsity of the inferred responses; varying this parameter leads to corresponding changes in the histograms of the deconvolved responses $\hat{s}_t$, with more or less probability mass assigned to $\hat{s}_t = 0$. Over a range of parameters, the ZIG model provides a good fit to the output histogram for all three of the algorithms examined here. We also found that the ZIG model provides a good fit to the output of deconvolution applied to data generated from an AR(2) model as well as the more biopysically detailed model from [108] (see Appendix

Figure 3.11).

Next, in Figure 3.3, we examine data shared through the SpikeFinder challenge [30], including traces recorded using four calcium indicators (GCamp6s, jRCAMP1a, OGB-1, jRGECO1a). Again, we find that the ZIG model provided a good fit across a wide range of data.

3.2.3 Constructing encoding models for simulated calcium responses

We have seen above that the ZIG model provides a good fit to the marginal distribution of the deconvolved responses $\hat{s}_t$. Now we want to exploit this probabilistic model to perform neural data analysis tasks. The first step is to fit *encoding models*: ie, what is $p(\hat{s}_t|\theta_t)$, for some observed covariate θ_t such as a stimulus or movement. In general, θ_t may be multi-dimensional, but in the example applications here θ_t will be one dimensional. Once these encoding models are fit and validated, we can use them to perform tasks like decoding of θ_t given the observed deconvolved responses $\hat{s}_t$. The overall approach is illustrated in Figure 3.4.

To fit the ZIG model to $p(\hat{s}_t|\theta_t)$, we need to fit three parameters, each of which may depend on θ: the probability of non-zero response $q(\theta)$, the scale parameter $a(\theta)$, and the shape parameter $k(\theta)$ for the gamma component, specifically

$$p\left(\hat{s}_t|\theta_t\right) = (1 - q(\theta_t)) \cdot \delta(0) + q(\theta_t) \cdot \text{gamma}\left(\hat{s}_t; k(\theta_t), a(\theta_t), loc\right), \tag{3.1}$$

where again we fix the location parameter loc for the gamma component as the minimum spike size s_{min}. (The mean of the ZIG model with parameters (q, k, a, loc) is $q(ka + loc)$, and the variance is $qka^2 + q(ka + loc)^2(1 - q)$.) We model these parameters as nonlinear functions of θ_t; we use neural networks to parameterize these nonlinearities, and then estimate the weights of these networks by maximum likelihood (see Section 3.4 for full details).

It is worth pausing to note two points here. First, the general problem of estimating

a mixture model whose parameters depend on θ_t would be rather challenging; however, in our case we are fitting a very particular two-component mixture model in which the first component (corresponding to $\delta(0)$) is trivial to estimate, since we merely need to count zero values in $\hat{s}_t$, resulting in a much easier estimation problem. Second, in the previous sections we showed that the ZIG provides a good *marginal* fit to $p(\hat{s}_t)$, marginalized over the whole dataset — but there is no guarantee that the same model provides a good *conditional* fit to $p(\hat{s}_t|\theta_t)$. We have checked this fit empirically, and it turns out that the ZIG model also provides a good fit to the conditional distributions considered here, in both real and simulated data (see Appendix Figure 3.12 for an example).

To test this conditional estimation approach, we generate artificial calcium imaging datasets with hundreds of simulated neurons (Figure 3.4). We first construct tuning curves of individual neurons that tile the space of θ values. In the real data examples presented below, θ will be a one-dimensional variable (e.g., the animal's head direction), so we use a one-dimensional θ in these simulations. Next we take a empirically measured time series θ_t (the head direction of a mouse), and compute the time-varying firing rates for individual neurons by plugging θ_t into the tuning curves. We then generate binned spike trains with different noise characteristics; we experiment with spike counts drawn from a Poisson distribution or a negative binomial (NB) distribution, as both have been proposed to model empirically observed spike responses [110, 111, 112]. Next we plug these simulated binned spike trains into the same generative model for calcium fluorescence traces discussed above, then deconvolve the resulting traces to obtain simulated responses $\hat{s}_t$. Finally, we fit the ZIG encoding model to the resulting responses $\hat{s}_t$.

We compare the ZIG model against simpler Poisson, Bernoulli, and gamma models (see Section 3.4 for full details). Figure 3.5 shows the results from a simulated dataset with negative binomial spiking. (We find that the results on the Poisson dataset are qualitatively similar; data not shown.) Overall, for both the Poisson or NB simulated datasets, we find that all of these models except for the Bernoulli model can capture the data mean well (the

Bernoulli model is only effective for data in which the mean of $\hat{s}_t$ in each bin is bounded below 1; this model fails to capture the responses in bins with high firing rates). However, only the ZIG model can properly capture both the mean and variability of $\hat{s}_t$. (The ZIG model also provides a good empirical fit to the full conditional distribution; data not shown.) The alternative models tend to either over- or under-estimate the variance, therefore providing poor descriptions of the distributions of the deconvolved responses; thus, the extra flexibility (due to the larger number of parameters) in the ZIG model is necessary to capture basic statistics of the data.

3.2.4 The ZIG encoding model leads to improved Bayesian decoding in simulated data

In the previous section we showed that the ZIG encoding model is flexible enough to capture the mean and variance of $\hat{s}_t$ across a wide range of firing rate regimes, in simulated data. Can we exploit this encoding model to obtain an improved decoder for θ_t? We use a classic Bayesian decoding approach to address this question: we compute the posterior distribution of θ_t, under the different encoding models for $\hat{s}_t$ discussed above, and then quantify how well the resulting posterior distributions capture the uncertainty in θ_t given the observed $\hat{s}_t$.

We quantify the performance of the resulting decoders on simulated data in Figure 3.6. Overall, the ZIG model leads to the smallest decoding error over a wide range of deconvolution sparsity parameters. Interestingly, accuracy degrades monotonically as a function of the sparsity of the output $\hat{s}_t$: i.e., the decoders can take advantage of even very small outputs $\hat{s}_t$ to improve the decoding accuracy. (In Figure 3.6 we use the deconvolution approach from [26], with a hard-threshold on the minimal spike size; results based on the soft-threshold deconvolution approach from [14] are similar.) The decoder based on the ZIG encoding model also achieves the highest coverage rate (i.e., the posterior credible interval covers the true value of θ_t with highest probability). In contrast, the decoders based on the Poisson and gamma models output credible intervals with mis-calibrated coverage rates (i.e., the credible

interval based on these models was narrower than it should have been), due to a mismatch between the true versus the modeled distribution of $\hat{s}_t$. In other words, a Bayesian statistician using a Poisson or gamma encoding model would be (mistakenly) overly confident in her predictions.

3.2.5 Application to real imaging data

In the previous section we developed the encoding-decoding analysis pipeline on simulated data. Next we apply these methods to real data. We focus on two calcium imaging datasets in this section. The first is a single-photon imaging dataset collected from thalamic region ADN, and the second is a two-photon dataset from hippocampal region CA1. Both datasets are collected in animals performing spatial navigation tasks (see Section 3.4 for full details). Our aim is to decode head direction (during free behavior) in the ADN data and location along a circular track (during head-fixed behavior) in the hippocampal data; thus in both cases the variable θ is one-dimensional, as in the simulated data. The proposed method has also been applied to decode the visual stimulus orientation [113].

We begin in Figure 3.7 by fitting encoding models to the ADN data (see Figure 3.8 for the results of hippocampal data). The results are similar to those shown in Figure 3.5: of the models examined here, only the ZIG model can capture both the mean and the variance of the empirical data. Further, in panel c we examine the tuning curves from this population of neurons. We compute the mean firing rates as a function of θ (leftmost panel) and plot these next to the estimated $a(\theta)$ and $q(\theta)$ curves (middle and right panels); recall that the mean of $\hat{s}$ as a function of θ scales proportionally with $a(\theta)q(\theta)$ in the ZIG model. We see that the parameters $a(\theta)$ and $q(\theta)$ covary across this population, indicating that there may be some statistical benefit in fitting these parameters with a hierarchical model that can share information between $a(\theta)$ and $q(\theta)$; however, we have not pursued this direction systematically.

Next we turn to decoding (Figures 3.9 and 3.10). Again, the results of the real data

analysis are largely consistent with the simulated results presented in Figure 3.6: in both datasets, the ZIG encoding model leads to more accurate Bayesian decoding, with higher credible interval coverage rates. Again, the decoding accuracy improves as the sparsity of $\hat{s}$ decreases.

One major difference between the simulated and real data is that the coverage probabilities are no longer well-calibrated, for any of the encoding models. In other words, the Bayesian posterior based on these encoding models is overly confident. We believe this is due to model mismatch: in our Bayesian decoder we model the responses $\hat{s}_t$ as conditionally independent across neurons and time given θ_t, i.e.,

$$p\left(\{\hat{s}_{ti}\}|\theta_t\right) = \prod_{it} p\left(\hat{s}_{ti}|\theta_t\right),$$

where $\hat{s}_{ti}$ denotes the observed response at time t from cell i. This assumption makes a testable prediction: $\hat{s}_{ti}$ should be uncorrelated with $\hat{s}_{tj}$ (where i and j index two different neurons) if we restrict attention to responses within a single bin of θ values. We find that these "noise correlations" are empirically not zero (invalidating the conditional independence assumption), and in fact if we perform a shuffling analysis in which we randomize the responses $\hat{s}_{ti}$ within each θ bin (thus preserving the relationship between $\hat{s}_{ti}$ and θ_t while destroying noise correlations between $\hat{s}_{ti}$ and $\hat{s}_{tj}$; see Section 3.4 for full details), then we find that the calibration of the credible interval is restored (data not shown). We leave further detailed modeling of these noise correlations to future work.

3.3 Discussion

The primary conclusion of this work is that the ZIG model provides a significantly improved fit to the distribution of the post-deconvolved calcium responses $\hat{s}_t$: the ZIG model is sufficiently flexible to capture the zero-inflation and varying mean and dispersion of the data across a wide variety of indicators, deconvolution methods, and behavioral settings. More-

over, it is straightforward to extend this into a θ_t-dependent encoding model, and in turn to use this encoding model for Bayesian decoding. The improved encoding fits provided by the ZIG lead directly to more accurate decoding, with better-calibrated posterior uncertainties. Finally, somewhat surprisingly, we find that setting the deconvolution hyperparameter to minimize the sparsity of $\hat{s}_t$ consistently leads to the most accurate decoder (consistent with results in [31]); i.e., attempting to discard small "noisy spikes" in $\hat{s}_t$ may be counterproductive. Overall, the ZIG model fills a crucial gap for calcium imaging analyses, by providing a firm statistical foundation for encoding and decoding models based on the estimated activity $\hat{s}_t$.

Of course the two-step approach followed here — to deconvolve the observed fluorescence traces, fit a probabilistic model to the deconvolved output, and then use this model to compute likelihoods in downstream encoding and decoding models — is not the only option. Another approach would be to use a full statistical model of the observed fluorescence traces (instead of treating the deconvolved output $\hat{s}_t$ as the observed data, as we did here), as in e.g. [115, 29, 28, 116]. This "end-to-end" modeling approach has the advantage that it can model more complex temporal dependencies in $\hat{s}_t$, and can potentially use side information to obtain better estimates of the neural activity from noisy fluorescence observations (see [22, 23, 117, 118, 14] for further examples along these lines). Conversely, there are a number of cases for which deconvolution or sophisticated statistical modeling is not required at all to address the scientific question at hand. The two-step approach pursued in this chapter can be seen as a useful compromise between these two extremes: if the researcher's scientific question requires more temporal resolution than is available from the raw fluorescence measurements (i.e., deconvolution is necessary), but the researcher lacks the time or expertise needed to develop, estimate, and test a full end-to-end statistical model, then the two-step approach developed here offers a quick, effective, practical compromise.

Open source code implementing the methods presented here along with the sample datasets is available at `https://github.com/zhd96/zig`. We hope these methods will be

useful for the variety of downstream analyses that are currently being pursued by the calcium imaging community.

3.4 Materials and Methods

3.4.1 Density models of the deconvolved calcium trace

ZIG model We model the density of the devonvolution output as

$$\hat{s}_{ti} \sim (1 - q_{ti}) \cdot \delta(0) + q_{ti} \cdot \text{gamma}(k_{ti}, a_{ti}, loc_i), \tag{3.2}$$

where q_{ti} denotes the probability of non-zeros, a_{ti} is the scale parameter of the gamma distribution, and k_{ti} is the shape parameter of the gamma distribution, for neuron i and time t. loc_i is the location parameter of the gamma distribution for neuron i, fixed as the minimum spike size s_{min}. We denote $a_i = (a_{1i}, \cdots, a_{Ti})^\top \in \mathbb{R}_+^T$ as the scale parameters for neuron i. Parameters q and k are defined similarly. We denote this family of density functions as $\hat{s} \sim \text{ZIG}(q, k, a)$. Note that when $s_{min} = 0$, the ZIG density family has a useful scale-invariance property: if $\hat{s} \sim \text{ZIG}(q, k, a)$, then $c\hat{s} \sim \text{ZIG}(q, k, ca)$, for any constant $c > 0$. This is convenient because in general the scale factor connecting spikes to increases in calcium concentration is unknown (and will typically vary from cell to cell); however, the scale invariance of the ZIG model implies that we do not need to estimate this scale factor explicitly.

(Scaled-)Poisson model The Poisson model places all of its probability mass on the non-negative integers, and is therefore inappropriate for modeling $\hat{s}_t$, which has range $\mathbb{R}_+$. Nonetheless, as discussed below, it is possible to assign a pseudolikelihood under the Poisson model to real-valued observations $\hat{s}_t$, and to fit the Poisson rate λ by maximizing this pseudo-likelihood. However, the Poisson model does not have the scale-invariance property enjoyed by the ZIG model, and therefore some care must be taken in defining a scale for

$\hat{s}_t$ (empirically, we find that the performance of the Poisson encoding and decoding models are highly sensitive to scaling of $\hat{s}_t$). We experiment with two scaling approaches. In the first scheme, the deconvolved trace is normalized by the noise standard deviation of the raw calcium trace, using methods proposed previously [14]. In the second approach, the deconvolved trace $\hat{s}_t$ is normalized by its Fano factor. Either of these normalizations leads to similar performance in terms of encoding or decoding accuracy (data not shown).

Bernoulli model The Bernoulli model can be considered as a special case of the ZIG model, by collapsing the positive responses into a delta function at 1. The responses are first binarized by thresholding; as discussed in the main text, we explore a range of different s_{min} values in the deconvolution step, and set the binarization threshold equal to s_{min}. As in the ZIG model, we define $q_i = (q_{1i}, \cdots, q_{Ti})^\top \in \mathbb{R}_+^T$ as the non-zero probability for neuron i.

Gamma model For completeness, we also fit a gamma model to the deconvolved responses. (Note that the gamma distribution can not capture the strong bimodality that we typically observe in $\hat{s}_t$.) The gamma distribution exhibits a singularity at 0 when the shape parameter k is less than 1. To avoid this issue, we slightly shift the observations away from 0 by adding a small positive number ϵ ($\epsilon = 10^{-4}$) to $\hat{s}$ before fitting the gamma(k, a) model. As in the ZIG model, we denote $a_i = (a_{1i}, \cdots, a_{Ti})^\top \in \mathbb{R}_+^T$ as the scale parameters for neuron i.

Parameter estimation We estimate the parameters of the above models via maximum (pseudo-)likelihood. Details appear in Section 3.4.3.

3.4.2 In vivo datasets

Two in vivo datasets are analyzed here. Both datasets are about 15 minutes long; in each case GCaMP6f was utilized as the calcium indicator. Traces are extracted using the CNMF-E software described in [19].

The first dataset is from area ADN of mouse thalamus. During stereotaxic surgery a male B6/C57j mouse was injected in ADN with the viral vector AAV9-hSyn-GCaMP6f (Molecular Tools Platform, Laval University). These mice were then implanted with a GRIN relay lens that was 500 microns in diameter and 4.0 mm in length (Inscopix, Inc.). The lens was positioned such that the bottom surface of the lens terminated just dorsal to the ADN. Baseplates used to attach the miniaturized fluorescent imaging endoscope ('UCLA Miniscope', miniscope.org) were cemented to the skull and imaging was performed using miniscopes while following the guidelines on the miniscope.org website. Recording sessions were conducted on a plus-maze (with each arm being 70cm long and 7.5 cm wide) in which animals were trained to alternate between arms. A webcam mounted above the maze tracked the position of a green and red light emitting diode that were attached to the miniscope. These were used to determine position and head direction of the mouse. Images were acquired at 30Hz. All experimental procedures followed the guidelines approved by the McGill University Animal Care Committee.

The second dataset is from area CA1 of mouse hippocampus. This dataset was collected using 2-photon imaging, while the head fixed male mouse was running for a stably placed hidden (non-cued) water reward on a 2 meter belt containing discrete tactile landmarks as in [119]. Images were acquired at 60Hz (post hoc temporally decimated to 30Hz).

3.4.3 Fitting encoding models to the data

We use a similar maximum likelihood-based fitting procedure for both the simulated data and the two real datasets. We denote $\theta = (\theta_1, \cdots, \theta_T)^\top \in (-180, 180]$. We split the data into 60% training data, 20% validation data, and 20% test data, for two simulation datasets and ADN data. For the CA1 data, only the data from running state are used, and the data are split it into 70% training data, 10% validation data, and 20% test data.

ZIG model There are three parameters, i.e., the scale parameter a and shape parameter k for the gamma component, and the probability of non-zero responses q. We parameterize the scale a and probability of non-zero responses q as a function of stimulus $(\sin(\theta), \cos(\theta))$ using neural networks, i.e.,

$$(a_{t1}, \cdots, a_{tN}, q_{t1}, \cdots, q_{tN}) = (f_1(\theta_t), \cdots, f_N(\theta_t), g_1(\theta_t), \cdots, g_N(\theta_t)),$$

where $f = (f_1, \cdots, f_N), g = (g_1, \cdots, g_N)$ are the output layer for a and q respectively. We use 2 hidden layers, each with tanh non-linearity, in the neural network. For the output layer, we use a logistic link function for f and an exponential link function for g. 30 nodes in hidden layers are used for the two simulated datasets and the ADN data; 15 nodes are used for the hippocampal data. We fix the shape parameter k to be a constant for individual neurons (i.e., k is neuron-dependent but not θ-dependent).

We optimize all the parameters by maximizing the log-likelihood using a variant of gradient descent, i.e., Adam [120]. Specifically, the objective function can be expressed as

$$\begin{aligned}\arg\max_{k,a,q} \sum_{i=1}^{N}\sum_{t=1}^{T} &\log(1-q_{ti})\,\mathbb{1}(\hat{s}_{ti}=0)+ \\ &\left(\log q_{ti} + (k_i-1)\log(\hat{s}_{ti}-s_{min}) - \frac{\hat{s}_{ti}-s_{min}}{a_{ti}} - k_i \log a_{ti} - \log\gamma(k_i)\right)\mathbb{1}(\hat{s}_{ti}>s_{min}).\end{aligned} \tag{3.3}$$

Poisson model We parameterize the Poisson mean λ using a neural network with the same structure as described for the ZIG model above, with an exponential link function in the output layer. Note that for the Poisson model, the likelihood function is not a proper likelihood because the the Poisson density can not be evaluated for non-integer values. We use a "psuedo-likelihood" function instead for the maximum likelihood estimation:

$$\arg\max_{\lambda} \sum_{i=1}^{N}\sum_{t=1}^{T} \hat{s}_{ti}\log\lambda_{ti} - \lambda_{ti}. \tag{3.4}$$

Bernoulli model The probability of positive response q is parameterized using a neural network with the same structure as in the ZIG model, with a logistic link function in the output layer. Formally, the objective function can be defined as

$$\arg\max_{q} \sum_{i=1}^{N} \sum_{t=1}^{T} (1 - \hat{s}_{ti}) \log (1 - q_{ti}) + \hat{s}_{ti} \log q_{ti}. \tag{3.5}$$

Gamma model The scale parameter a and the shape parameter k are parameterized with the same neural network structure as in the ZIG model, except using an exponential link in the output layer. The objective function can be expressed as

$$\arg\max_{k,a} \sum_{i=1}^{N} \sum_{t=1}^{T} (k_i - 1) \log \hat{s}_{ti} - \frac{\hat{s}_{ti}}{a_{ti}} - k_i \log a_{ti} - \log \gamma(k_i). \tag{3.6}$$

3.4.4 Shuffling analysis

In section 3.2.5 we performed a shuffling analysis to investigate the conditional independence assumption used by the Bayesian decoder. Details of this analysis are provided here. We started with the original T-by-N matrix $\hat{s}_{ti}$ (with T denoting the number of observed video frames, and N the number of extracted cells), then made a new matrix $\tilde{s}_{ti}$ as follows. For each time point t and each cell i, we randomly chose a timestep u (u depends on (t, i)) such that $\theta_t = \theta_u$, and then set $\tilde{s}_{ti} = \hat{s}_{ui}$. The new matrix $\tilde{s}$ has the same marginal distribution of $p(\hat{s}_t|\theta_t)$, so the encoding models will be the same, but the correlations between cells will be destroyed.

3.5 Appendix

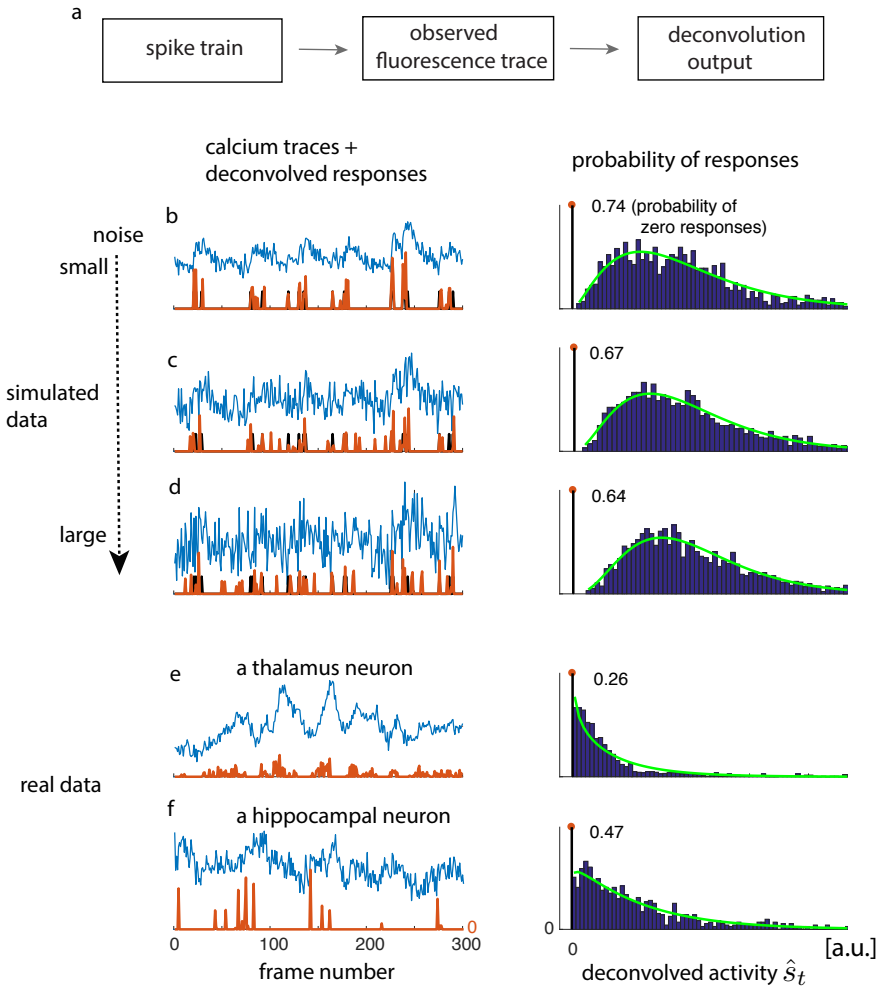

Figure 3.1: Illustration of the zero-inflated gamma (ZIG) model: deconvolved calcium responses typically consist of a mixture of zero responses plus a continuous component that is well-modeled as a gamma distribution, in both simulated and real imaging data. (a) Pipeline of the simulations for the generation of artificial data. The spike train is sampled from a inhomogeneous Poisson process. The calcium concentration is then determined by a autoregressive process (AR(1) process, with decay time constant 450ms), driven by the spike train. The observed calcium trace is determined by the calcium concentration plus independent Gaussian noise. The deconvolved calcium responses are obtained using the OASIS deconvolution algorithm described in [26]. (b,c,d) Left: observed fluorescence trace (blue), ground truth spikes (black), and the deconvolved output (orange). Each frame = 30ms. Right: the histogram of the deconvolved output (blue) and the ZIG fit (green); the number on each histogram represents the proportion of zero responses. The additive noise level of the simulated fluorescence increases from panel b to panel d. (e) Observed fluorescence and deconvolved response of a neuron from ADN (see Section 3.4 for full experimental details). (f) Same as panel e but from the hippocampus (again, see Section 3.4 for full experimental details). (Conventions as in b-d but we no longer have access to the true spikes.) In each case, the ZIG model provides a good fit to the deconvolved outputs.

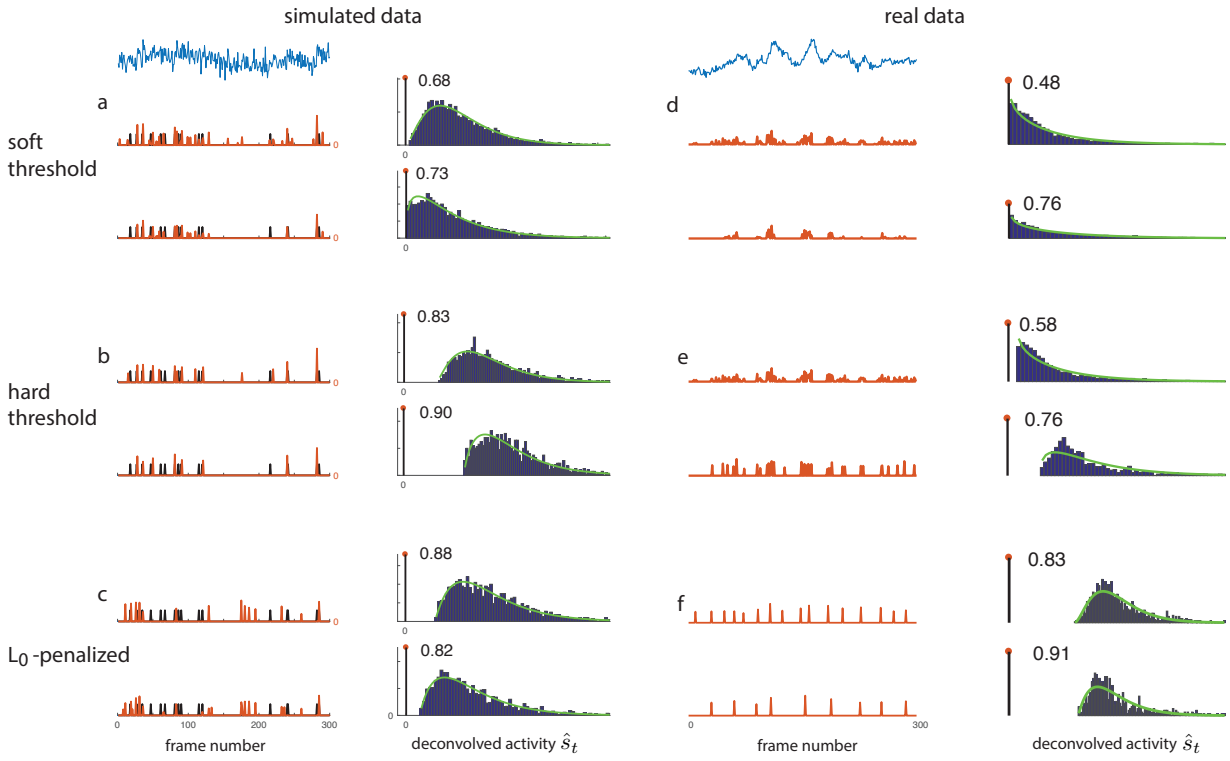

Figure 3.2: The ZIG model is robust with respect to the details of different deconvolution methods. We consider three deconvolution methods, including a L_1-penalization method with soft threshold (panel a,d) [14], a method with non-zero minimal spike size or hard threshold (panel b,e) [26], and an L_0-penalized method (panel c,f) [27]. We apply these methods to both simulated (panel a,b,c) and real data (panel d,e,f; same traces as in Figure 3.1). Each method has a hyper-parameter controlling the sparseness of the deconvolved activity. Two values of the sparseness parameter are examined for each method. Conventions as in Figure 3.1.

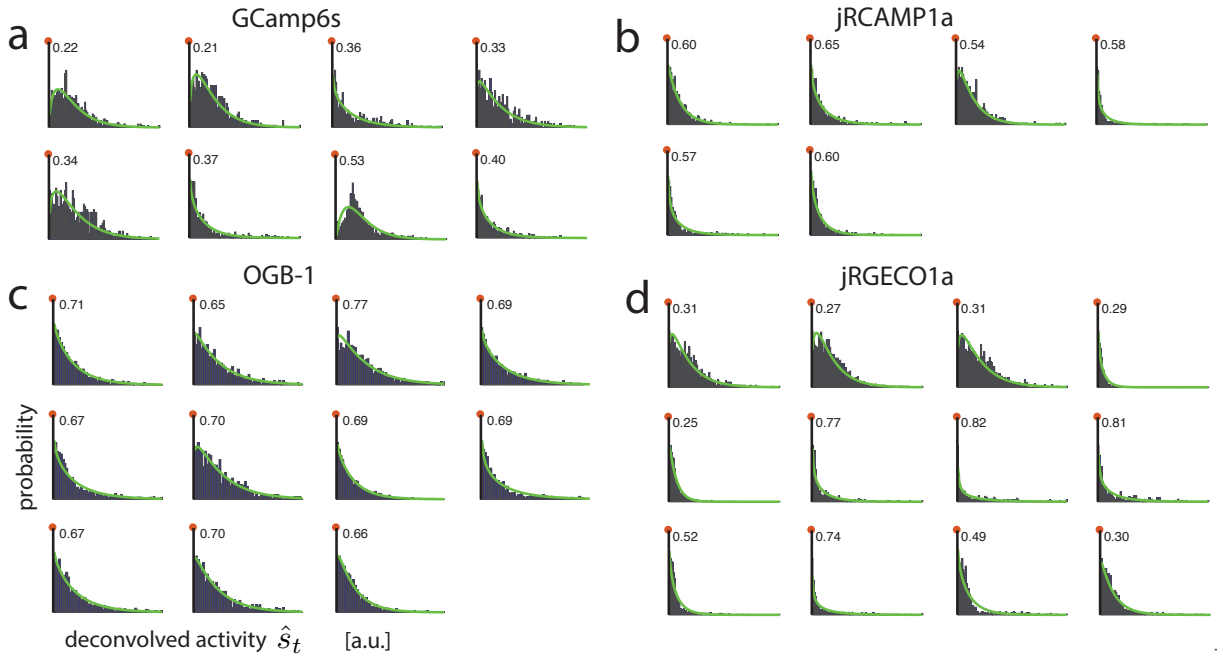

Figure 3.3: The ZIG model is robust with respect to data collected with different calcium indicators. Data collected using four different calcium indicators are tested. The data are from the SpikeFinder challenge dataset (panel a,c from [25]; panel b,d from [109]). (Specifically, the four datasets used here are SpikeFinder dataset #1 (calcium indicator OGB-1), dataset #3 and #5 (indicator GCamp6s), dataset #9 (indicator jRCAMP1a), and dataset #10 (indicator jRGECO1a).) All the neurons examined here fired at least 200 spikes.

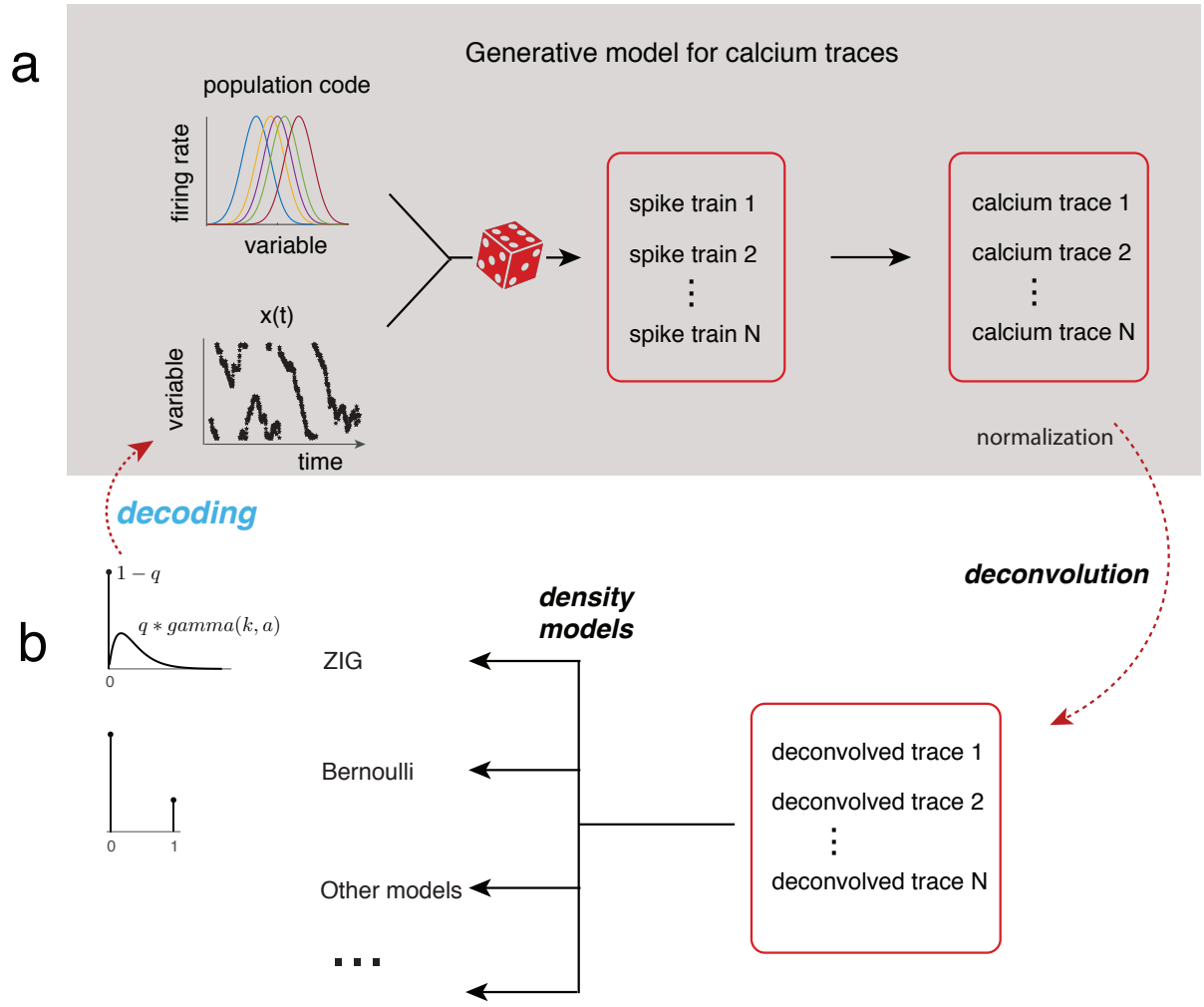

Figure 3.4: The encoding-decoding modeling framework. (a) The full generative "encoding" model for calcium fluorescence traces. Given the neural tuning curves and the covariate sequence θ_t, spikes are generated probabilistically and transformed (as in Figure 3.1) into the observed fluorescence traces. (b) After deconvolving to obtain $\hat{s}_t$, we can use the ZIG as well other models for $\hat{s}_t$, and use the estimated encoding model and $\hat{s}_t$ to decode θ_t.

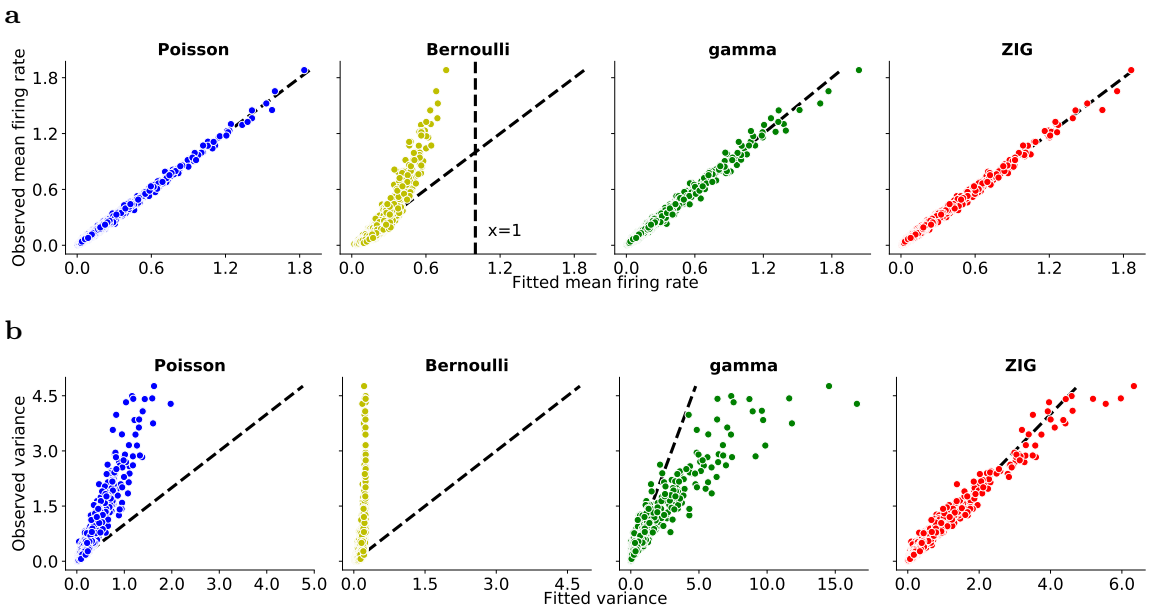

Figure 3.5: The ZIG model captures the means and variances of deconvolved calcium responses in the simulated data. (a,b) Summary plots based on all the neurons ($N = 215$) showing the observed versus the predicted mean/variance of $\hat{s}_t$. We divide the θ range to into small equi-spaced bins (number of bins = 18 here), and compute the mean and variance for the observed and fitted responses corresponding to each bin for each model neuron. Each dot represents the mean (or variance) associated with one bin from one neuron.

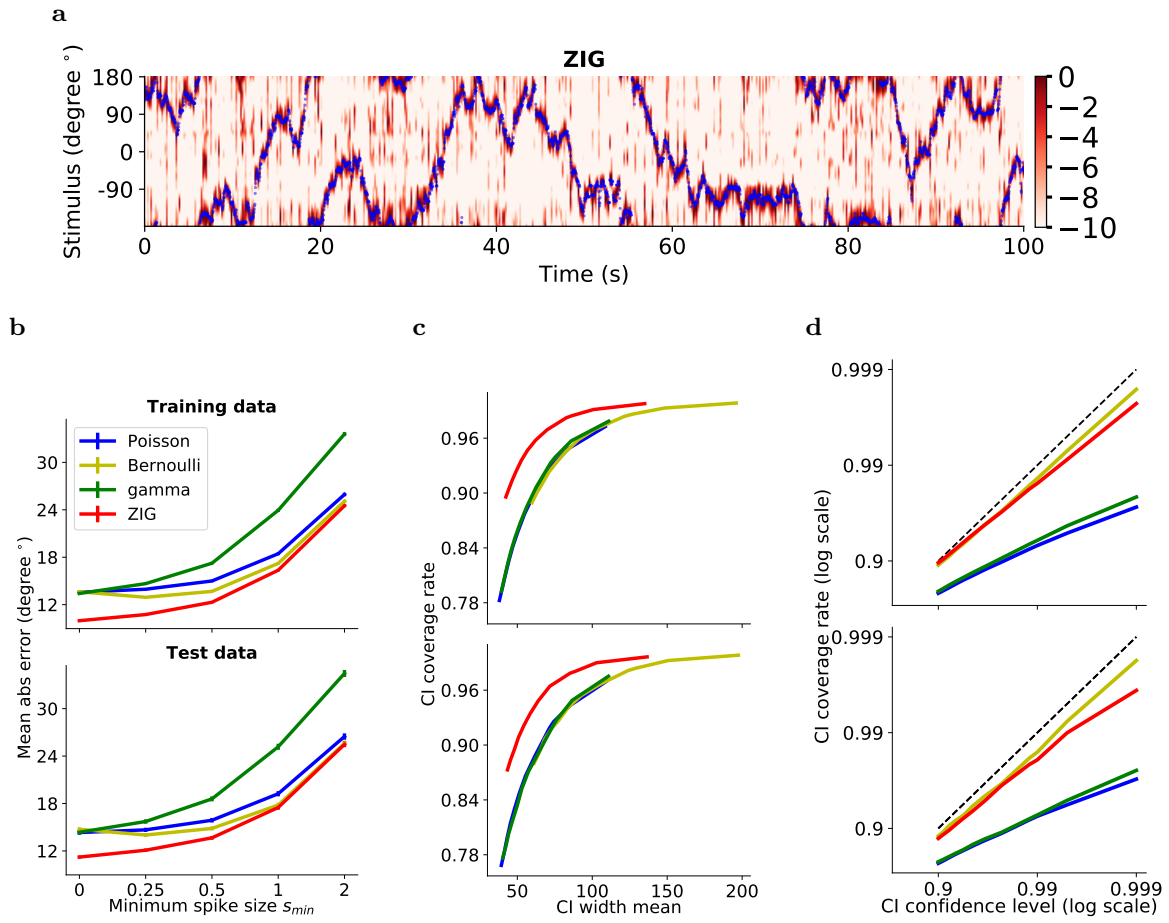

Figure 3.6: The ZIG model leads to improved decoding performance on simulated data. Decoding is performed based on the deconvolved responses in single frame, time window $\sim$ 33ms, with no smoothing across frames on all neurons ($N = 215$). (a) True simulated location (blue) plotted with the decoded normalized log-posterior probability under the ZIG model (red; posterior at each step is normalized to have a maximum of 1, for easier visualization). Note that the decoded posterior does a good job of tracking the true location. (b) Decoding mean absolute error ± 1 standard error under different encoding models, with varying s_{min}, the minimum spike size parameter in [26]; larger values of s_{min} correspond to sparser output $\hat{s}_t$. (c) Posterior credible interval (CI) width vs CI coverage rate (the probability that the true location falls within the CI; higher is better here). (d) Confidence level vs CI coverage rate. Dashed line indicates unity (i.e., the CI is achieving its nominal coverage rate). In (b,c) we see that the ZIG model leads to the lowest decoding error and the best coverage rate over a range of parameters, while (d) shows that the CI computed under the ZIG model achieves a nearly-nominal coverage rate, as desired; in contrast, the Poisson and gamma encoding models output mis-calibrated credible intervals.

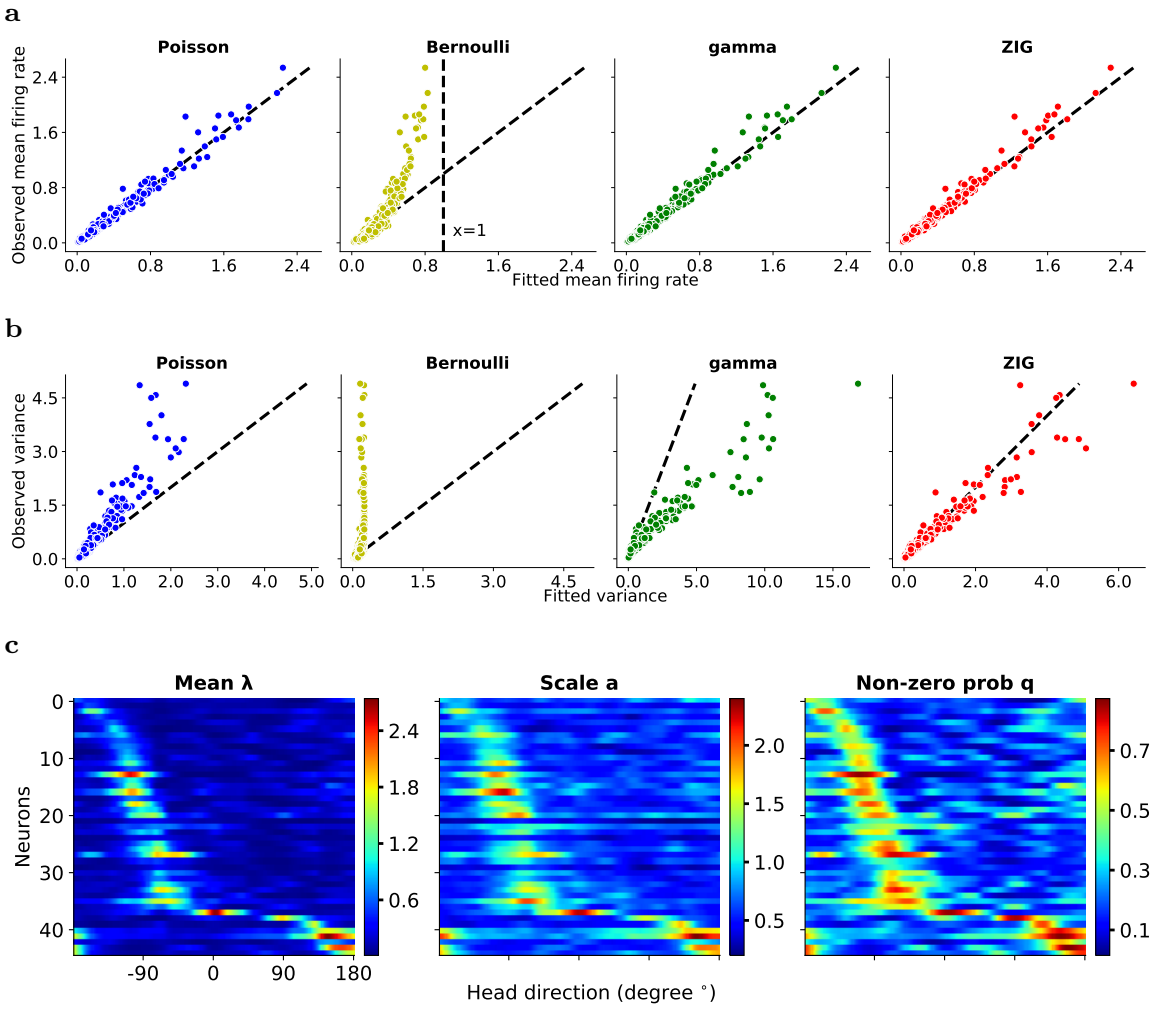

Figure 3.7: The ZIG model best captures the means and variances of deconvolved calcium responses in the ADN dataset. (a,b) Conventions as in Figure 3.5; similar to results based on the simulated data, the ZIG model again provides the best fits to the observed means and variances for this experimental dataset. (c) Estimated parameter values for the ZIG model and Poisson model, for multiple neurons (each row corresponds to one neuron, while the columns correspond to different θ values). Neurons are sorted according to the preferred firing direction. For the Poisson model, each row plots the mean firing rate λ as a function of head direction for each neuron. For the ZIG model, there are three sets of parameters. The scale parameter a and the probability of non-zero response q are plotted. Notice that the two parameters are correlated; both parameters tend to scale with the estimated mean rate λ. We find that we could obtain good fits by fixing the shape parameter k in the ZIG model for each neuron.

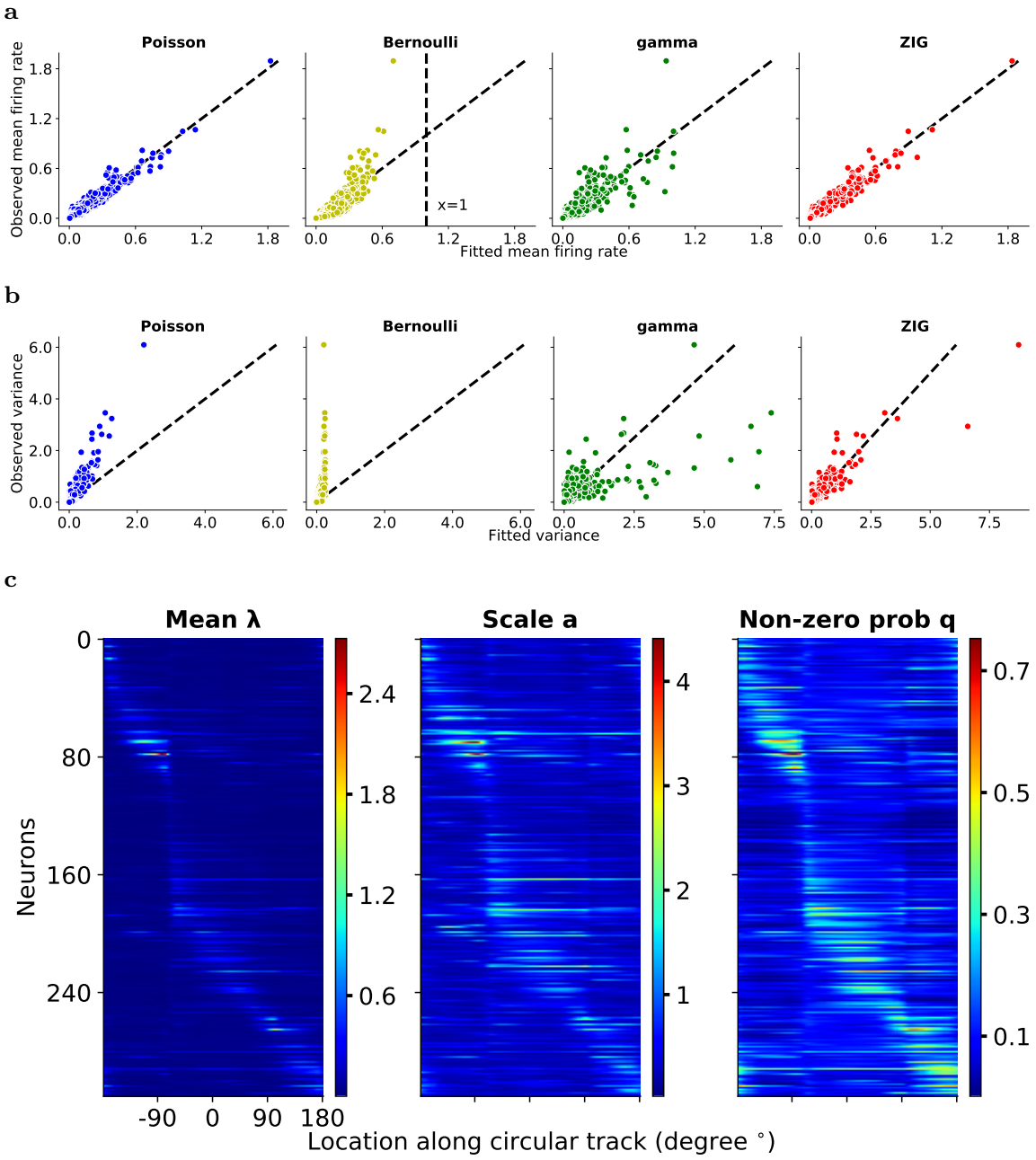

Figure 3.8: Encoding results for hippocampus data during running. Conventions as in Figure 3.7. (a,b) The observed versus the predicted mean/variance of $\hat{s}_t$. (c) Estimated parameter values for the ZIG model and Poisson model, for multiple neurons. Again, among the four models considered, only the ZIG model can capture both the mean and variance of the calcium responses conditional on the animal's location on the track.

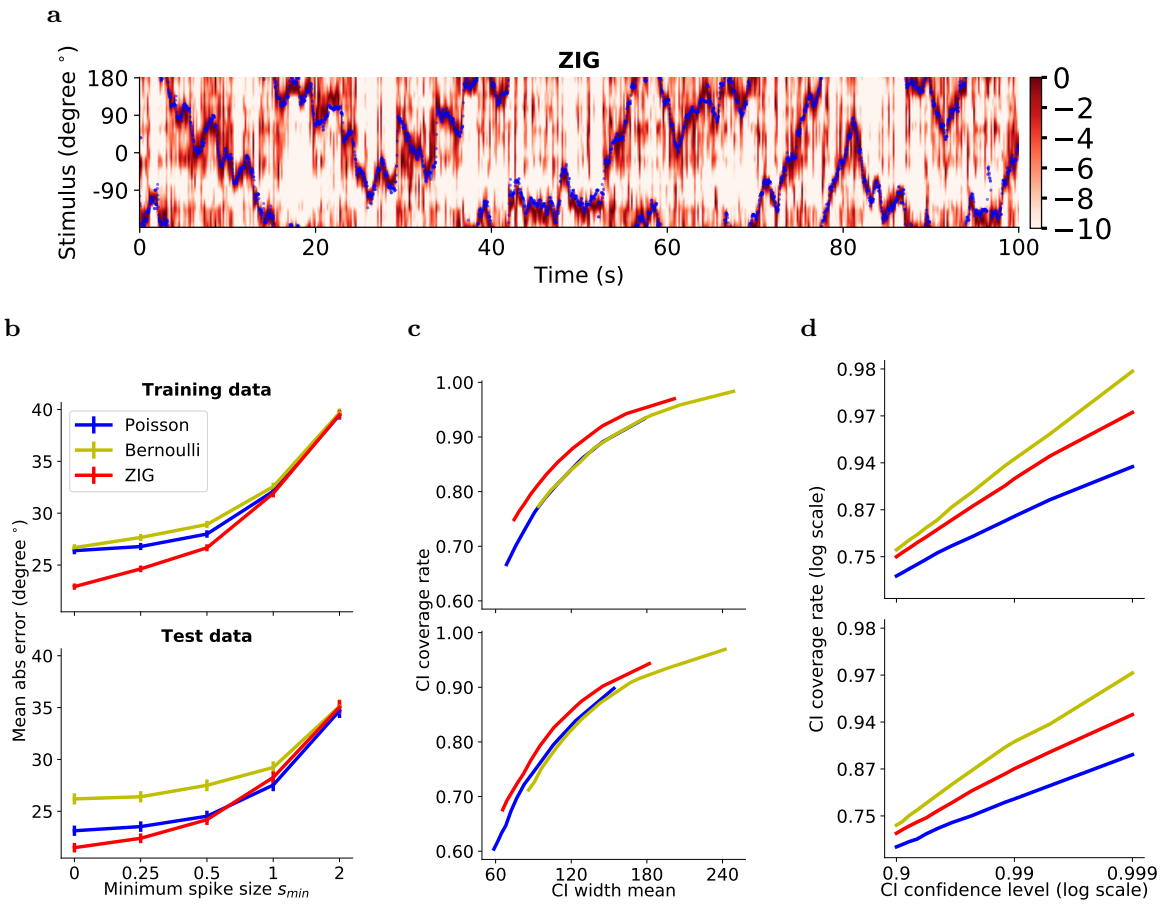

Figure 3.9: Decoding results for head direction (ADN) dataset. Decoding is performed based on responses from single frame, time window $\sim$ 33ms, with no smoothing across frames on $N = 45$ neurons. Conventions similar to Figure 3.6. (a) True simulated location (blue) plotted with the decoded normalized log-posterior probability for the ZIG model (red). (b) Decoding mean absolute error ± 1 standard error under different encoding models, with varying s_{min}. (c) Posterior CI width vs CI coverage rate. (d) Confidence level vs CI coverage rate. The gamma model performed poorly here and is not shown.

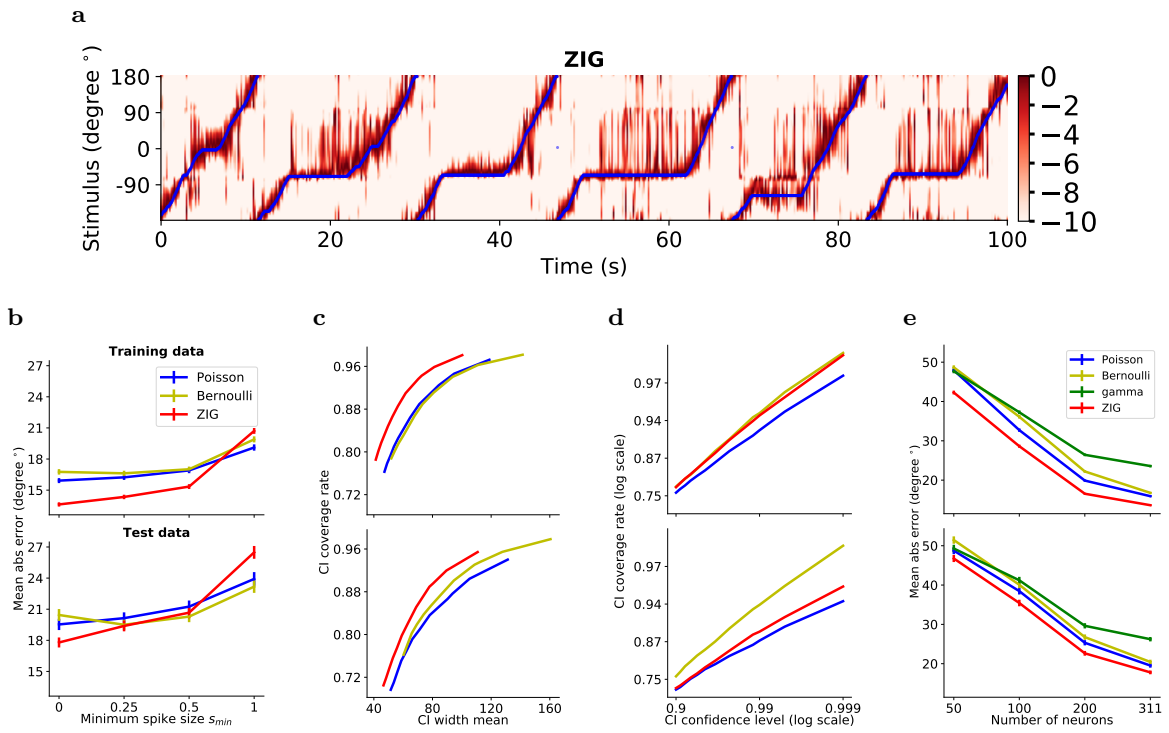

Figure 3.10: Decoding results for hippocampus data. The encoding model is estimated based on the data during running, as is typical in the hippocampal decoding literature [114]. Decoding time window is set to be every two frames, $\sim$ 33ms. (a,b,c,d) similar to Figure 3.9. (e) Decoding results for subsets of neurons of hippocampus data. We randomly select neurons ($N = 50, 100, 200$) out of the 311 total neurons and perform the encoding-decoding analysis based on these subsets. The advantage of the ZIG model remains robust with smaller sub-populations of hippocampal neurons.

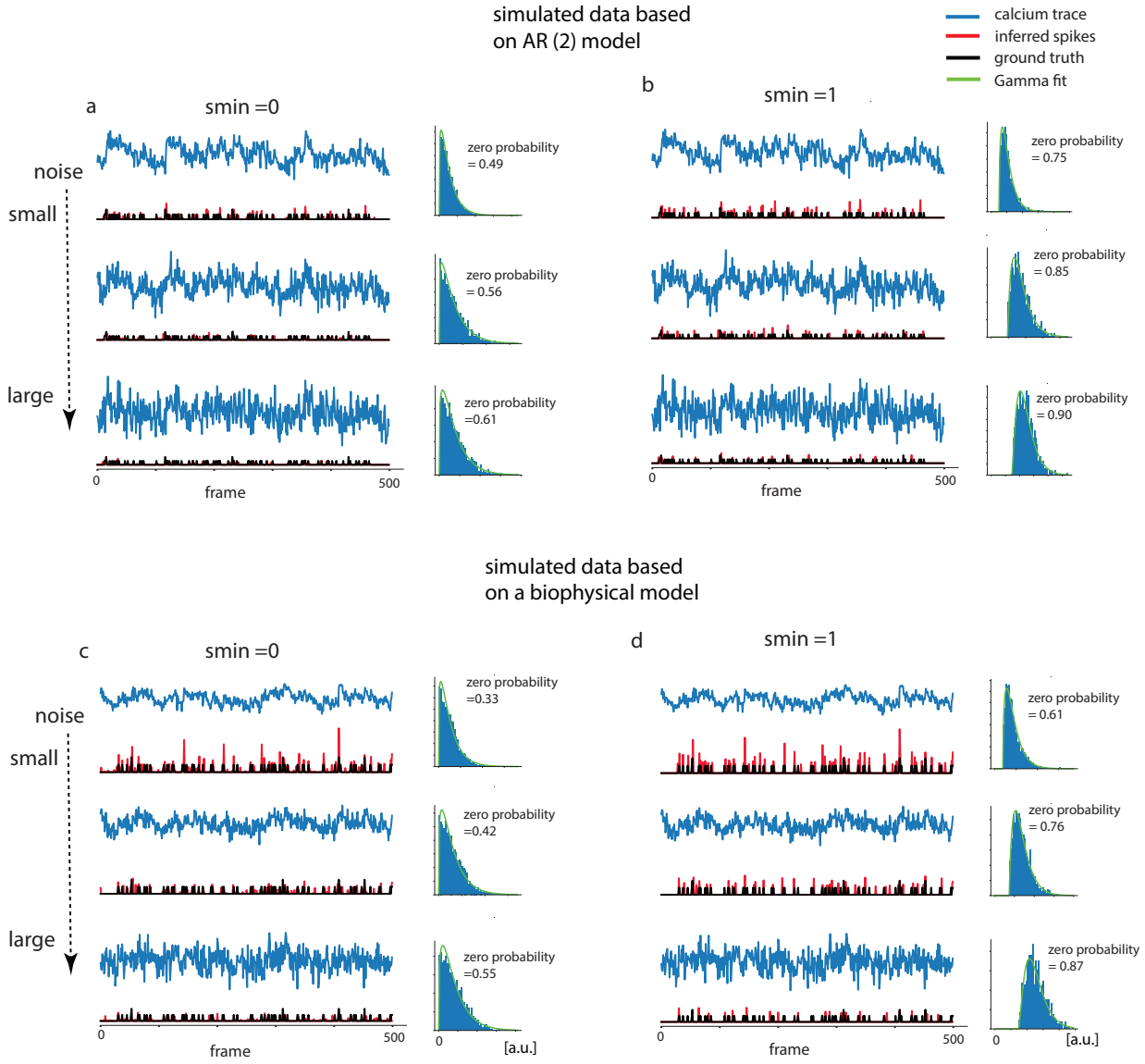

Figure 3.11: The ZIG model provides a good fit to the deconvolved calcium responses of simulated data based on a AR(2) model and a more biophysical realistic model. The spike train is sampled from a homogeneous Poisson process. (a,b) Results based on an AR(2) model under three different noise levels and two different smin values in deconvolution [23, 14, 26]. Left: fluorescence trace from simulations (blue), ground truth spikes(black), and the deconvolved output (red). Right: the histogram of the positive deconvolved output (blue) and the ZIG fit (green). (c,d) Similar to (a,b), but based on a more biophysical realistic model as described in [108].

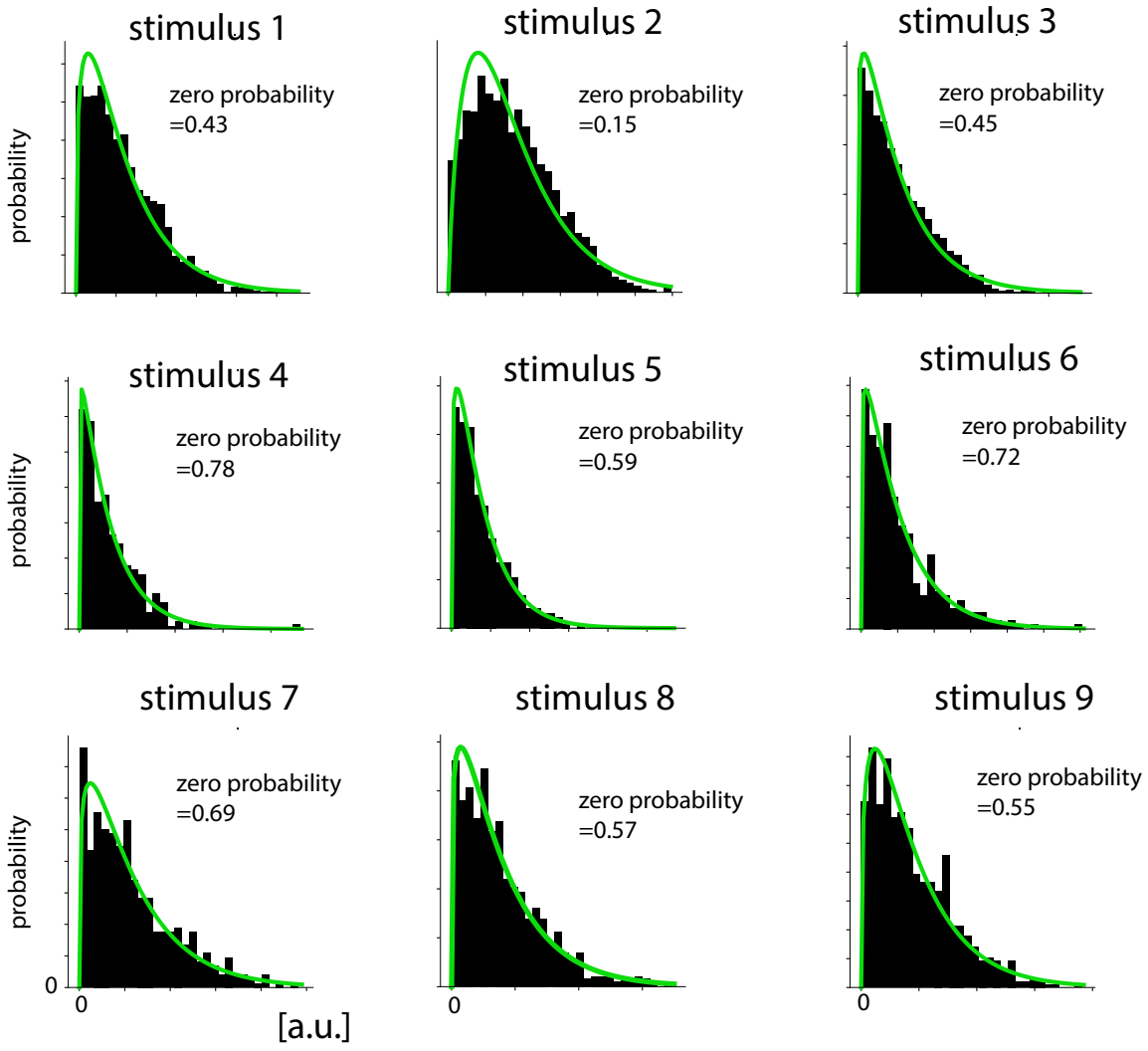

Figure 3.12: The ZIG model provides a good fit to the conditional probability distribution of deconvolved calcium responses given stimulus. The spike train is sampled from an inhomogeneous Poisson process with firing rate modulated by the stimulus, and the calcium concentration is determined by an AR(1) model. Each panel shows the histogram of the positive deconvolved output corresponds to each stimulus value. The green line is the ZIG fit.

Chapter 4: Disentangled sticky hierarchical Dirichlet process hidden Markov model

This chapter was published as "Disentangled sticky hierarchical Dirichlet process hidden Markov model" [85] in The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD) 2020 with Yuanjun Gao and Liam Paninski. We thank John Paisley, Xue-Xin Wei, Kenneth Kay, Matthew R Whiteway, Pengcheng Zhou, and Ari Pakman for helpful discussions. We thank the authors of [121] for generously sharing their data. We acknowledge computing resources from Columbia University's Shared Research Computing Facility project, which is supported by NIH Research Facility Improvement Grant 1G20RR030893-01, and associated funds from the New York State Empire State Development, Division of Science Technology and Innovation (NYSTAR) Contract C090171, both awarded April 15, 2010.

4.1 Introduction

Hidden Markov models (HMMs) provide a powerful set of tools for modeling time series data. In the HMM we assume that the time series observations are modulated by underlying latent time-varying variables which take a discrete set of states. This model class is useful in its own right and can also be incorporated as a building block for more complicated models. It has been widely used in speech recognition [122], musical audio analysis [123, 124], acoustic-phonetic modeling [125], behavior segmentation [126, 127, 121], sequential text modeling [68, 128], financial time series data analysis [126, 129], computational biology [130], and many other fields.

Selecting the number of HMM states is an important question for practitioners. Clas-

sical model selection techniques can be used, but these methods can be computationally intensive and are sometimes unreliable in practice [131, 132]. Also, for real datasets, it is often reasonable to assume that the number of latent states may be unbounded, violating classical assumptions needed to establish consistency results for model selection. Based on previous work in [133], [68] proposed the Hierarchical Dirichlet process HMM (HDP-HMM), a Bayesian nonparametric framework. In the HDP-HMM the transition matrix follows a hierarchical Dirichlet process (HDP) prior. [122] noted that the HDP-HMM tends to rapidly switch among redundant states, and proposed the sticky HDP-HMM (S-HDP-HMM), which strengthens the self-persistence probability. This modification often leads to significant improvements in modeling real data.

In the HMM, it is important to distinguish three features: 1, the similarity of the rows of the transition matrix; 2, the average self-persistence probability of the latent states (controlled by the mean of the diagonal of the transition matrix); and 3, the strength of the self-persistence prior across states (i.e., the inverse prior variance of the diagonal elements of the transition matrix). In the HDP-HMM, there is only one parameter controlling feature 1. The sticky HDP-HMM adds one more parameter to control feature 2, but still entangles features 1 and 3 with only one parameter, thus limiting the expressiveness of the prior.

We show that we can add one additional hyperparameter to generalize the sticky HDP-HMM formulation, obtaining three degrees of freedom to model the three features discussed above. We call this new model the disentangled sticky HDP-HMM (DS-HDP-HMM).

The rest of the chapter is organized as follows. In section 4.2, we provide a brief introduction to Bayesian HMM, HDP-HMM, and sticky HDP-HMM. In section 4.3, we discuss the limitations of these models. In section 4.4, we introduce disentangled sticky HDP-HMM, and in section 4.5 we develop efficient Gibbs sampling inference methods for this new model. Section 4.6 demonstrates the effectiveness of the disentangled sticky HDP-HMM on both synthetic and real data, including applications to analyzing neural data and segmenting behavior video. The notation table can be found in the Appendix 4.8.1.

4.2 Background on Bayesian HMM and HDP-HMM

Our goal here is to fit an HMM to time series data. On its face, this would seem to be a solved problem; after all, we can compute the HMM likelihood easily, and the basic expectation-maximization algorithm for HMM fitting is textbook material [64]. Nonetheless, a fully Bayesian solution to this problem has remained elusive. Specifically, we would like to be able to compute a posterior over all of the unknown HMM parameters (including the number of latent states). Quantification of posterior uncertainty is critical in many applications: for example, given short time series data, often we do not have enough data to sufficiently identify the HMM parameters. Even for longer time series data, we might want to fit richer models as we collect more data. Here "richer models" correspond to more latent states, and since the number of parameters in the HMM grows quadratically with the number of states, we may be left again with some irreducible uncertainty about the model parameters.

The HDP-HMM [68] provides a useful starting point for fully Bayesian HMM inference. The basic idea here is to sample a global transition distribution prior from a Dirichlet process (described below), and then for each latent state we sample a transition distribution from this shared (random) global prior distribution. To develop the details of this HDP-HMM idea we first need to define some notation for the Dirichlet process (DP). Given a base distribution H on a parameter space Θ and a positive concentration parameter γ, a Dirichlet process $G \sim \mathrm{DP}(\gamma, H)$ (sometimes also denoted by $\mathrm{DP}(\gamma H)$) can be constructed by the following stick-breaking procedure [134]: let

$$\beta \sim \mathrm{GEM}(\gamma),\ \theta_i \overset{iid}{\sim} H,\ i = 1, 2, \cdots, \tag{4.1}$$

where $\beta \sim \mathrm{GEM}(\gamma)$ is a random probability mass function (p.m.f.) defined on a countably

infinite set as follows:

$$v_i \sim \text{Beta}(1, \gamma),\ \beta_i = v_i \prod_{l=1}^{i-1}(1 - v_l),\ i = 1, 2, \cdots . \tag{4.2}$$

Then the discrete random measure $G = \sum_i \beta_i \delta_{\theta_i}$ is a sample from $\text{DP}(\gamma H)$, where δ_{θ_i} denotes the Dirac measure centered on θ_i.

The HDP-HMM [68] uses the DP to define a prior on the rows of the HMM transition matrix in a setting where the number of latent states is unbounded. The HDP-HMM is defined as

$$\begin{aligned}
\text{DP shared global prior}: &\quad \beta \sim \text{GEM}(\gamma) \\
&\quad \theta_j \overset{iid}{\sim} H,\ j = 1, 2, \cdots \\
\text{Transition matrix prior}: &\quad \pi_j \overset{iid}{\sim} \text{DP}(\alpha\beta),\ j = 1, 2, \cdots \\
\text{Latent states}: &\quad z_t \sim \pi_{z_{t-1}},\ t = 1, \cdots, T \\
\text{Observations}: &\quad y_t \sim f(y|\theta_{z_t}),\ t = 1, \cdots, T
\end{aligned} \tag{4.3}$$

Here, β and $\{\theta_j\}_{j=1}^{\infty}$ are defined as in the DP described above, and then each transition distribution π_j for state j is defined as a random sample of a second DP with base measure β and concentration parameter α. Here α controls how similar π_j is to the global transition distribution β. Finally, as usual, z_t denotes the state of a Markov chain at time t, and the observation y_t is independently distributed given the latent state z_t and parameters $\{\theta_j\}_{j=1}^{\infty}$, with emission distribution $f(\cdot)$.

The sticky HDP-HMM from [122] modifies the transition matrix prior by adding a point mass distribution with stickiness parameter κ to encourage self-persistence:

$$\text{Transition matrix prior}: \quad \pi_j \sim \text{DP}(\alpha\beta + \kappa\delta_j),\ j = 1, 2, \cdots , \tag{4.4}$$

where δ_j denotes the Dirac measure centered on j. Figure 4.1(a) provides the graphical model for the sticky HDP-HMM.

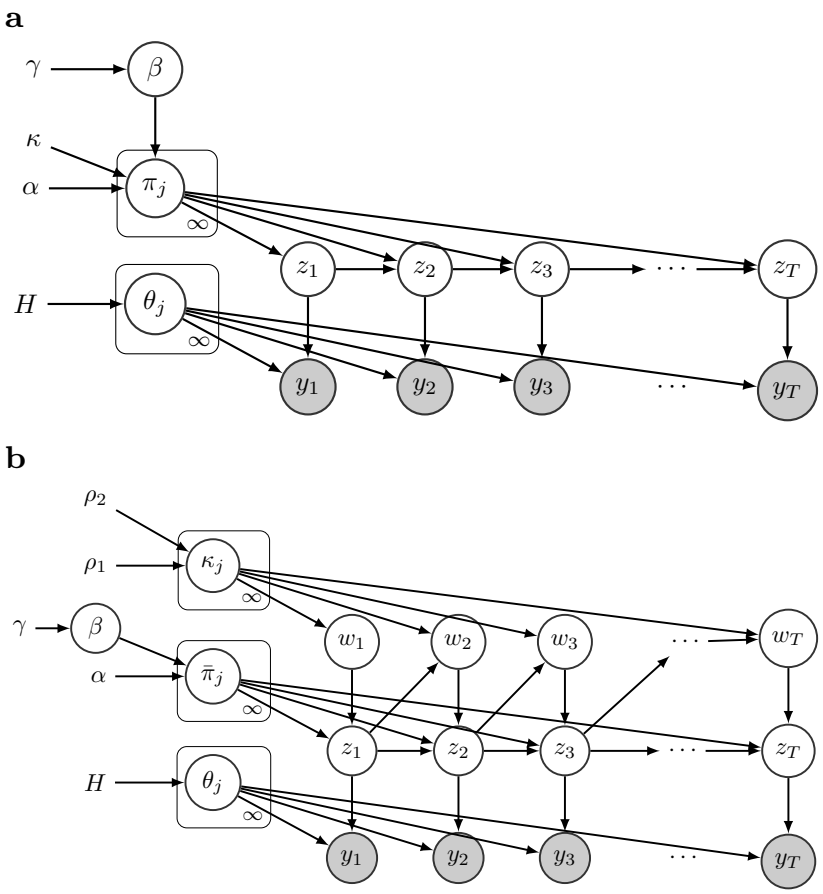

Figure 4.1: Graphical models for sticky HDP-HMM (a) and disentangled sticky HDP-HMM (b).

4.3 Limitations of the HDP-HMM and sticky HDP-HMM

The HDP-HMM uses the concentration parameter α to control the following feature of the HMM:

- feature 1: the strength of the transition matrix prior, or the similarity of the rows of the transition matrix.

In other words, a large value of α here means that the transition probability for each state is close to the global transition distribution β.

A flexible model should have additional parameters to control two additional features:

- feature 2: the average self-persistence probability, or the mean of the diagonal of the

transition matrix.

- feature 3: the strength of the self-persistence prior, or the similarity of the diagonal elements of the transition matrix.

The sticky HDP-HMM adds just one parameter κ compared to the HDP-HMM. feature 2 is controlled by $\kappa/(\alpha+\kappa)$, while both feature 1 and feature 3 are controlled by $\alpha+\kappa$. Note that a strong prior (large $\alpha+\kappa$) means that both the self-persistence probability and the transition probability are quite similar across states, and it is impossible to separate the strength of these two elements using this parameterization. These three features should occupy three degrees of freedom in total, but the formulation of sticky HDP-HMM is only able to traverse a two-dimensional sub-manifold of this three-dimensional space, limiting the expressiveness of the sticky HDP-HMM prior.

More concretely, consider the speaker diarization example studied in [122]. The task is to distinguish the speakers in an audio recording of a conversation; the current speaker is the hidden state. Suppose that some speakers are likely to speak for a very long time while others are terse (implying a small feature 3, i.e. small $\alpha+\kappa$), but the identity of the next speaker is independent of the identity of the previous speaker (implying a big feature 1, i.e. big $\alpha+\kappa$). The sticky HDP-HMM would have trouble in this scenario. The opposite case is plausible as well: as an example, consider a group of people sitting in a circle and expressing their opinions in a clock-wise fashion (implying a small feature 1 since the distribution of the next speaker is highly dependent on the previous speaker); if each speaker talks for a very similar amount of time (implying a big feature 3), then the sticky HDP-HMM would have difficulty with this scenario as well.

4.4 Disentangled sticky HDP-HMM

Now that we have diagnosed this lack of flexibility in the HDP-HMM and sticky HDP-HMM, we can construct a new more flexible model that separates the strength of the self-

persistence from the similarity of the transition probabilities. Specifically, we modify the transition matrix prior as

$$\begin{aligned} \text{Transition matrix prior}: \quad & \kappa_j \overset{iid}{\sim} \text{beta}(\rho_1, \rho_2) \\ & \bar{\pi}_j \overset{iid}{\sim} \text{DP}(\alpha\beta) \\ & \pi_j = \kappa_j \delta_j + (1-\kappa_j)\bar{\pi}_j, \ j = 1, 2, \cdots, \end{aligned} \tag{4.5}$$

where the transition distribution π_j is a mixture distribution. A sample from π_j has the self-persistence probability of κ_j to come from a point mass distribution at j, and has probability $1-\kappa_j$ to come from $\bar{\pi}_j$, a sample from the DP with base measure β. We call this new model the disentangled sticky HDP-HMM.

Here the beta(ρ_1, ρ_2) prior has the flexibility to control both the expectation of self-persistence (feature 2), and the variability of self-persistence (feature 3). Meanwhile α is free to control the variability of the transition probability around the mean transition β (feature 1). In short, we use 3 parameters (ρ_1, ρ_2, α) rather than the 2 parameters (κ, α) to separate the strength of the self-persistence and the transition priors.

When $\rho_1 = 0$ and $\rho_2 > 0$, all the $\kappa_j = 0$, and the disentangled sticky HDP-HMM reduces to HDP-HMM. Importantly, the sticky HDP-HMM is also a special case of the disentangled sticky HDP-HMM, as shown in Theorem 1. See the Appendix 4.8.2 for a proof.

Theorem 1. *The sticky HDP-HMM formulation in equation 4.4 is a special case of the disentangled sticky HDP-HMM by setting* $(\rho_1, \rho_2) = (\kappa, \alpha)$.

An equivalent formulation of $z_t \sim \pi_{z_{t-1}}$ in the disentangled sticky HDP-HMM is as follows:

$$\begin{aligned} \text{Latent states}: \quad & w_t \sim \text{Ber}(\kappa_{z_{t-1}}) \\ & z_t \sim w_t \delta_{z_{t-1}} + (1-w_t)\bar{\pi}_{z_{t-1}}, \ t = 1, \cdots, T, \end{aligned} \tag{4.6}$$

where we add binary auxiliary variables w_t, which decide whether the next step is self-persistent or switching. See Figure 4.1(b) for the corresponding graphical model. We use

this formulation to facilitate inference in section 4.5.

4.5 Gibbs sampling inference

In this section, we introduce a new direct assignment Gibbs sampler (Algorithm 4.5.1) (similar to [68]) and a new weak-limit Gibbs sampler (Algorithm 4.5.2) (similar to [122]) for inference in the disentangled sticky HDP-HMM. The direct assignment sampler generates samples from the true posterior of the disentangled sticky HDP-HMM when the Gibbs chains converge. The weak-limit sampler uses finite approximation of the HDP-HMM to accelerate the mixing rate of the Gibbs chains and can be easily adapted to parallel computing. [126] noted that the weak-limit sampler was useful for observation models with dynamics such as auto-regressive HMM (ARHMM) or switching linear dynamic system (SLDS). For detailed derivations of these two algorithms, see the Appendix 4.8.3.

4.5.1 Direct assignment sampler

The direct assignment sampler for the HDP-HMM marginalizes transition distributions π_j and parameters θ_j and sequentially samples z_t given all the other states $z_{\backslash t}$, observations $\{y_t\}_{t=1}^T$, and the global transition distribution β. The main difference between our direct assignment sampler and the corresponding HDP-HMM sampler is that instead of only sampling z_t, we sample $\{z_t, w_t, w_{t+1}\}$ in blocks. We sample α, β, γ only using z_t that switch to other states by $\bar{\pi}_{z_{t-1}}$ $(w_t = 0)$, and sample $\{\kappa_j\}_{j=1}^{K+1}, \rho_1, \rho_2$ only using z_t that stick to state z_{t-1} $(w_t = 1)$.

Algorithm 3 Direct assignment sampler for disentangled sticky HDP-HMM

1: Sequentially sample $\{z_t, w_t, w_{t+1}\}$ for $t = 1, \cdots, T$.
2: Sample $\{\kappa_j\}_{j=1}^{K+1}$. K is defined as number of unique states in $\{z_t\}_{t=1}^T$.
3: Sample β. Same as HDP-HMM.
4: Optionally, sample hyperparameter $\alpha, \gamma, \rho_1, \rho_2$.

For step 1, we sequentially compute the probability for each possible case of the posterior $p(z_t, w_t, w_{t+1} | z_{\backslash t}, w_{\backslash \{t,t+1\}}, \{y_t\}_{t=1}^T, \alpha, \beta, \{\kappa_j\}_{j=1}^{K+1})$, and sample $\{z_t, w_t, w_{t+1}\}$ together from the

corresponding multinomial distribution. If $z_t = K + 1$, i.e. a new state appears, we will increment K, sample self-persistence probability κ_{K+1} for a new state from the prior, and update β using stick-breaking. For step 2, given w_{t+1} whose corresponding z_t is j, we can sample κ_j using beta-binomial conjugacy. For step 3, by introducing auxiliary variables $\{m_{jk}\}_{j,k=1}^{K}$, we sample β using Dirichlet-multinomial conjugacy. For step 4, we compute the empirical transition matrix $\{n_{jk}\}_{j,k=1}^{K}$, where n_{jk} is the number of transitions from state j to k with $w_t = 0$ in $\{z_t\}_{t=1}^{T}$, and introduce additional auxiliary variables. Then the posterior of α and γ are gamma-conjugate, given the auxiliary variables. We approximate the posterior of ρ_1, ρ_2 by finite grids. The complexity for each step in Algorithm 4.5.1 is $\mathcal{O}(TK)$, $\mathcal{O}(K)$, $\mathcal{O}(K)$, and $\mathcal{O}(K)$ respectively, so the total complexity per iteration is $\mathcal{O}(TK)$.

It is worth noting that instead of modeling $\{\kappa_j\}_{j=1}^{K+1}$ as samples from a beta distribution, it is natural to consider any distribution on the $[0, 1]$ interval. The Gibbs algorithm here is easily adaptable to cases where we have extra prior information on the self-persistence probability.

4.5.2 Weak-limit sampler

The weak-limit sampler for the sticky HDP-HMM constructs a finite approximation to the HDP prior based on the fact that

$$\begin{aligned} \beta|\gamma &\sim \mathrm{Dir}\left(\gamma/L, \cdots, \gamma/L\right) \\ \pi_j|\alpha, \beta &\sim \mathrm{Dir}\left(\alpha\beta_1, \cdots, \alpha\beta_L\right), \ j = 1, \cdots, L \end{aligned} \tag{4.7}$$

converges to the HDP prior when L goes to infinity. Using this approximation, one can jointly sample latent variables $\{z_t\}_{t=1}^{T}$ with the HMM forward-backward procedure [64], which accelerates the mixing rate of the Gibbs sampler.

The main difference between our weak-limit Gibbs sampler and the corresponding sticky HDP-HMM sampler is that we now have two dimensional latent variables $\{z_t, w_t\}_{t=1}^{T}$ to sample.

Algorithm 4 Weak-limit sampler for disentangled sticky HDP-HMM

1: Jointly sample $\{z_t, w_t\}_{t=1}^T$.
2: Sample $\{\kappa_j\}_{j=1}^L$.
3: Sample $\{\beta_j\}_{j=1}^L, \{\bar{\pi}_j\}_{j=1}^L$. Same as HDP-HMM.
4: Sample $\{\theta_j\}_{j=1}^L$.
5: Optionally, sample hyperparameter $\alpha, \gamma, \rho_1, \rho_2$.

For step 1, we apply the forward-backward procedure to jointly sample the two dimensional latent variables $\{z_t, w_t\}_{t=1}^T$. Step 2 is the same as in Algorithm 4.5.1. For step 3, we sample β and $\bar{\pi}$ based on Dirichlet-multinomial conjugacy, given auxiliary variables $\{m_{jk}\}_{j,k=1}^L$, the empirical transition matrix $\{n_{jk}\}_{j,k=1}^L$, and the approximate prior in equation 4.7. For step 4, we place a conjugate prior on θ_j and use conjugacy to sample from the posterior. Step 5 is the same as in Algorithm 4.5.1. The complexity for each step in Algorithm 4.5.2 is $\mathcal{O}(TL^2)$, $\mathcal{O}(L)$, $\mathcal{O}(L)$, $\mathcal{O}(L)$, and $\mathcal{O}(L)$ respectively, with total complexity $\mathcal{O}(TL^2)$.

By jointly sampling the full latent sequence $\{z_t, w_t\}_{t=1}^T$, the weak-limit sampler greatly improves the mixing rate. The correlated observations in ARHMM and SLDS further slows the mixing rate of the direct assignment sampler, so jointly sampling is especially important for those models with dynamics [126].

4.6 Results

In this section, we apply the disentangled sticky HDP-HMM to both simulated and real data. We compared the performance of our disentangled sticky HDP-HMM with the two baseline models: sticky HDP-HMM and HDP-HMM. We evaluated the model performance according to two metrics: normalized Hamming distance (defined as the element-wise difference between the inferred states and the true underlying states) on training data, and the predictive negative log-likelihood on held-out test data. To compute the Hamming distance, we used the Munkres algorithm [135] to map the indices of the estimated state sequence to the set of indices that maximize the overlap with the true sequence. To compute the

predictive negative log-likelihood, we used the set of parameters inferred every 10th Gibbs iteration after it converges, and ran the forward algorithm of [64].

For all the experiments, we adopted a full Bayesian approach, and used the following hyperpriors. For all three models, we placed a Gamma$(1, 0.01)$ prior on the concentration parameters α (and for the sticky HDP-HMM on $\alpha + \kappa$) to cover a wide range of α values. We placed a Gamma$(2, 1)$ prior on γ to avoid extremely small and large γ samples, because small γ will cause numerical instability when sampling β, while large γ will generate too many states. For our model and sticky HDP-HMM, we placed non-informative priors on the self-persistence parameters. We placed a Unif$([0, 1])$ prior on the self-persistence proportion parameter $\phi = \frac{\rho_1}{\rho_1+\rho_2}$. For our model, we placed a Unif$([0, 2])$ on the self-persistence scale parameter $\eta = (\rho_1 + \rho_2)^{-1/3}$, and cut $[0, 1] \times [0, 2]$ (the support of ϕ, η) into 100×100 grids (for simulated data) or 30×30 (for real data) to numerically compute the posterior for ϕ, η.

We used the direct assignment sampler to fit the simulated data, because the simulated data here has a relatively small sample size, and the direct assignment sampler generates samples from the true posterior of the model. We used the weak-limit sampler to fit real data (with a larger sample size) in parallel. We have also applied the direct assignment sampler to a short version of the hippocampal data; the results obtained are qualitatively similar.

4.6.1 Simulated data

In the simulation studies, we focus on two settings that serve to clearly illustrate the differences between the sticky versus disentangled sticky models. In both settings, we experimented with both multinomial and Gaussian emissions. See the Gaussian emission results in the Appendix 4.8.5.

Different self-persistence, same transition

We simulated data using a transition matrix (Figure 4.2(a)) with different κ_j and the same $\bar{\pi}_j$ across states, which corresponds to a large transition concentration parameter α

(big feature 1), and small self-persistence parameters ρ_1, ρ_2 (small feature 3). We assigned the multinomial observation to be equal to the latent state with probability 0.9 and to other states with equal chance with probability 0.1.

We placed a symmetric $\text{Dir}(1, \cdots, 1)$ prior on the multinomial parameters. We ran 3 MCMC chains, each with 15000 iterations, with 11000 iterations as burn-in (Figure 4.2(d)).

Results of three models fit on multinomial emission are shown in Figure 4.2. As shown in Figure 4.2(e), our model learns a big α and small ρ_1, ρ_2, which is consistent with the data, while sticky HDP-HMM model entangles these two parameters and learns something in the middle. Better recovery of the hyperparameters leads to better fits on the data: our model outperforms the two baseline models in terms of negative log-likelihood and Hamming distance (Figure 4.2(b)(c)). The advantages of our model are even more clear under the Gaussian emission (see figure in the Appendix 4.8.5). The three models generally learn similar γ, which means that they infer similar numbers of states.

For each emission model, we compared these three models on 10 datasets generated from the same model using different random seeds. The conclusions are consistent across the 10 different datasets.

Same self-persistence, different transition

We simulated data using a transition matrix (Figure 4.3(a)) with the same κ_j and different $\bar{\pi}_j$ across states, which corresponds to small α (small feature 1), and large ρ_1, ρ_2 (large feature 3). We assigned the multinomial observation to be equal to the latent state with probability 0.8 and to other states with equal chance with probability 0.2.

Again, consistently across 10 replications, our model learns hyperparameters consistent with the data (Figure 4.3(d)) and outperforms the two baseline models (Figure 4.3(b)(c)), though the advantages are not as big as in the previous scenario. This is likely because in the previous example, our model can learn the hyperparameters based on the similarity among rows of the transition matrix, while in this scenario, it can learn hyperparameters mostly

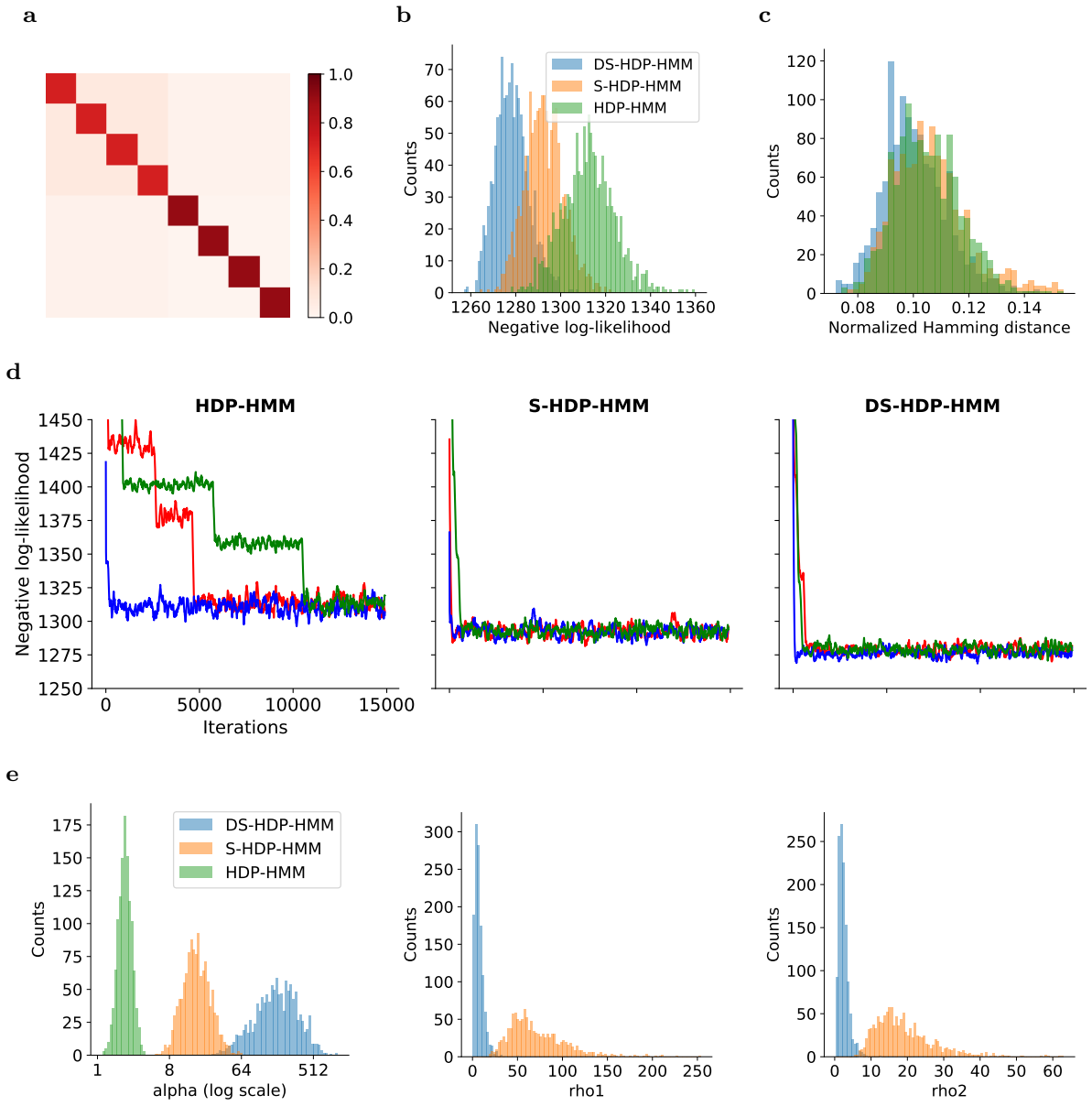

Figure 4.2: The DS-HDP-HMM provides good fits to the simulated data with different self-persistence (ρ_1, ρ_2 small), same transition (α large), and multinomial emission. (a) True transition matrix. (b) Histogram of negative log-likelihood on test data. (c) Histogram of normalized Hamming distance between the estimated states and true states on training data. (d) Negative log-likelihood on test data over 15000 direct assignment Gibbs samples from 3 chains for each model. (e) Histogram of hyperparameters α, ρ_1, ρ_2. Note that we plot κ, α for sticky HDP-HMM in the histogram of ρ_1, ρ_2, since κ, α would be equal to ρ_1, ρ_2 respectively if we treat sticky HDP-HMM as a special case of our model. Our model learns a big α and small ρ_1, ρ_2, which is consistent with the data.

from the similarity among diagonal elements of the transition matrix, which contain much less information.

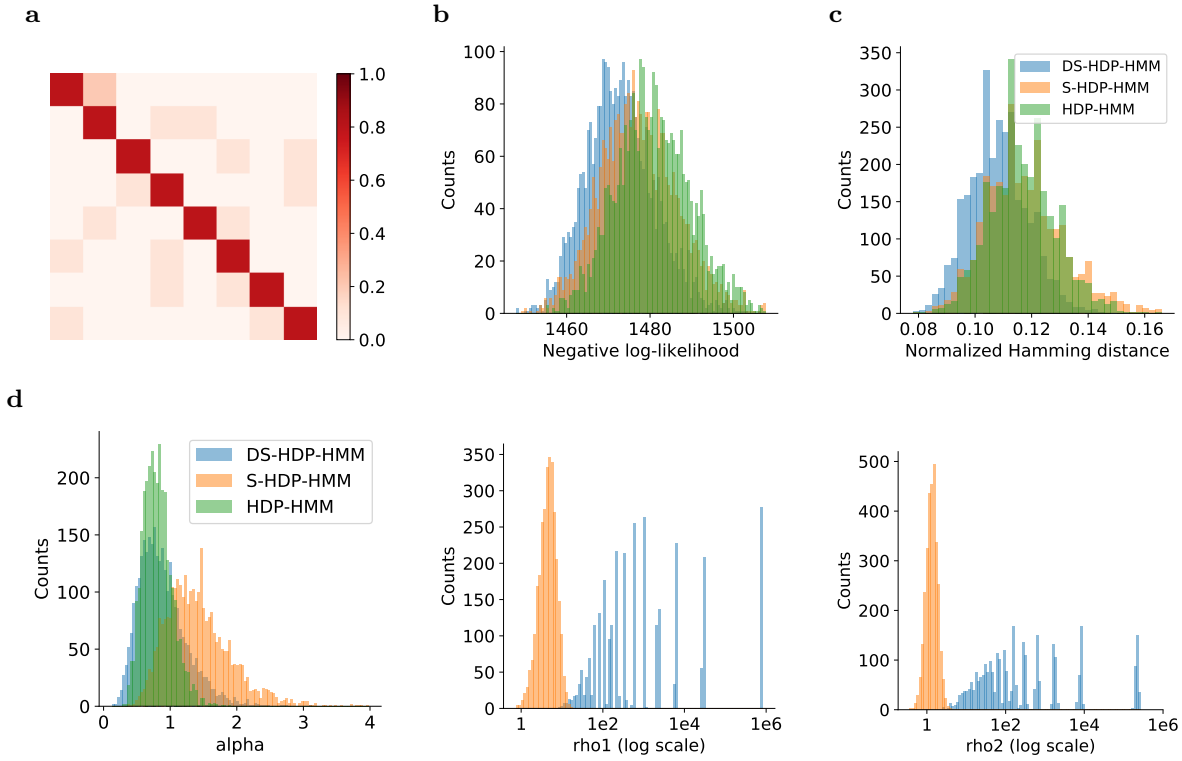

Figure 4.3: Results for the simulated data with same self-persistence (ρ_1, ρ_2 large), different transition (α small) and multinomial emission. We ran 3 chains, each with 30000 iterations, with 20000 iterations as burn-in for this data. Conventions as in Figure 4.2. Our model learns a small α and big ρ_1, ρ_2, which is consistent with the data. Note that the "spikiness" in the histogram of ρ_1, ρ_2 comes from the approximation of the posterior of ρ_1, ρ_2 using finite grids. An alternative way to avoid the discretization would be to use Metropolis-Hasting sampling [77].

4.6.2 Inferring rat hippocampal population codes

Next we applied our model to a public electrophysiology hippocampus dataset [136, 137][1]. In the experiment, a rat freely explored in an open square environment ($\sim$ 50cm$\times$40cm), while neural activity in the hippocampal CA1 area was recorded using silicon probes. See

[1] `https://buzsakilab.nyumc.org/datasets/PastalkovaE/i01/i01_maze15_MS.001/`

Figure 4.4(a) for an example trace illustrating the position of the rat over the course of the experiment.

We selected the 100 most active putative pyramidal neurons and binned the ensemble spike activities with a frame rate of 10Hz. The dataset consists of $\sim$ 36k frames. We used the first 8k frames, cut it into blocks of 500 frames, and randomly took 8 blocks as training data (4k frames), and the remaining 8 blocks as test data. The spike count at time t for cell c is modeled as Poisson with rate $\lambda_{z_t,c}$, i.e. $y_{t,c} \sim \text{Poisson}(\lambda_{z_t,c})$. As in [138], we used a conjugate gamma(a_c, b_c) prior for the firing rate $\lambda_{j,c},\ j = 1, 2, \cdots,\ c = 1, \cdots, 100$. We fixed the shape parameter $a_c = 1$ across cells, and placed a gamma$(1, 1)$ prior for the scale parameter b_c.

We set $L = 200$ and ran 7 MCMC chains, each with 15000 iterations, with 11000 iterations as burn-in (Figure 4.4(d)). Our model achieves the smallest negative log-likelihood on test data (Figure 4.4(c)). A two-sample t-test for the mean difference between negative log-likelihood of DS-HDP-HMM and S-HDP-HMM for 7 chains is significant (p-value < 0.05). The inferred states are correlated with the spatial locations of the rat (Figure 4.4(b)). Compared to sticky HDP-HMM, our model infers a bigger α and smaller ρ_1, ρ_2 (Figure 4.4(e)). Bigger α implies smaller variability of the switching transition, consistent with the rat quickly going to other locations since it has fast running speed. Smaller ρ_1, ρ_2 imply bigger variability of the self-persistence probability, consistent with the rat spending different durations at different locations. Note that the disentangled sticky HDP-HMM and sticky HDP-HMM perform similarly on the rat hippocampal data; bigger differences are seen in the mouse behavior video data in the next section.

4.6.3 Segmenting mouse behavior video

We also applied our model to a public mouse behavior dataset [139, 140]. In this experiment, a head-fixed mouse performed a visual decision task while neural activity across dorsal cortex was optically recorded using widefield calcium imaging. We only used the

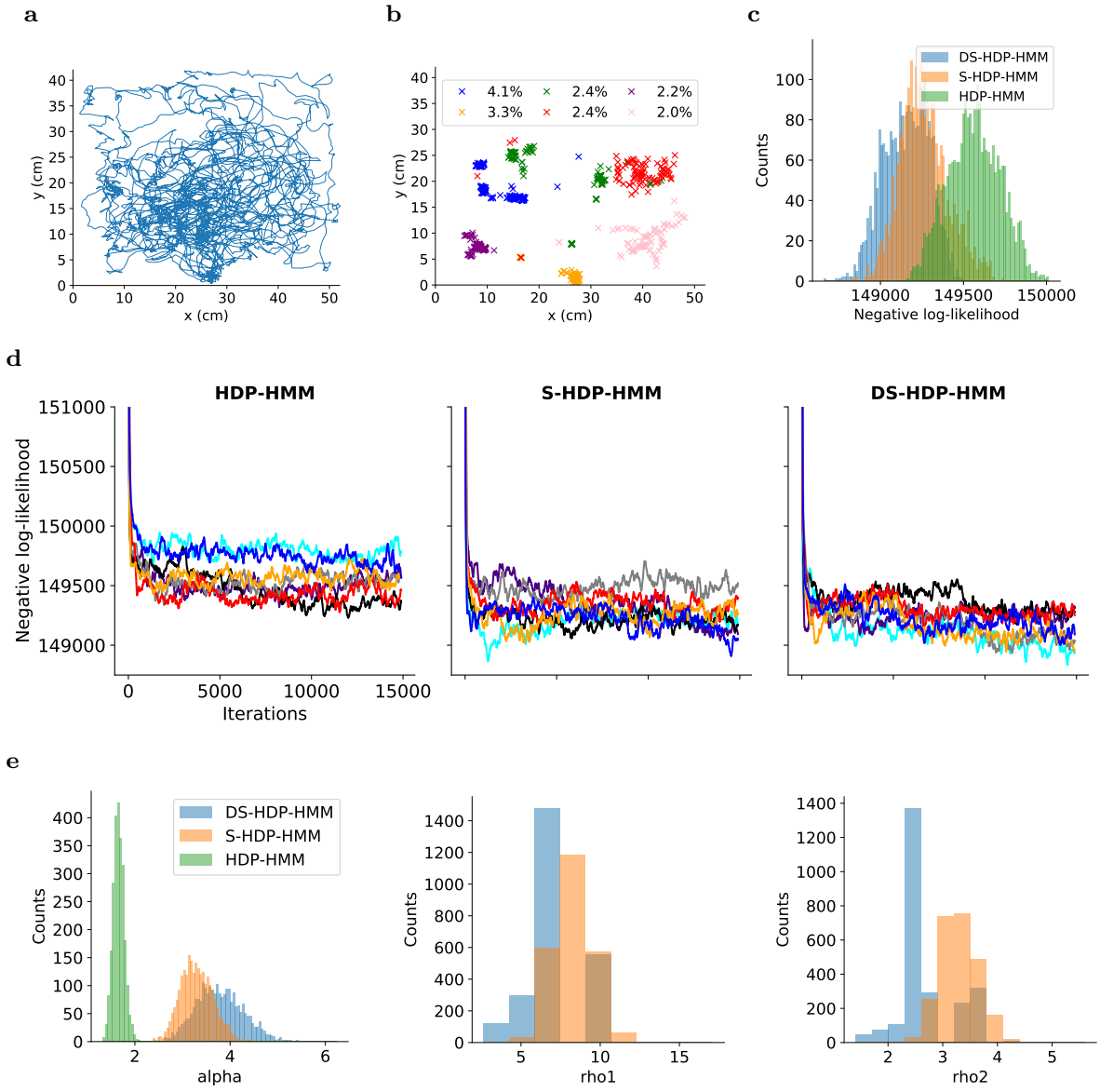

Figure 4.4: Results for rat hippocampal data. (a) The rat's moving trajectory in training and test data. (b) The spatial locations of the rat corresponding to 6 most frequent states inferred by DS-HDP-HMM. The figure legend shows the percentage of each of the 6 states. (c)(d)(e) Conventions as Figure 4.2, though with more traces and using the weak-limit sampler in (d). Our model infers a bigger α than ρ_1, ρ_2 (big feature 1, small feature 3).

behavior video data (128x128 pixels each grayscale video frame), which was recorded using two cameras (one side view and one bottom view). The behavior video is high dimensional, so we directly adopted the dimension reduction result in [121], which output 9-dimensional

continuous variables estimated using a convolutional autoencoder.

The dataset consists of 1126 trials across two sessions, with 189 frames each trial (30 Hz frame rate). We randomly chose 100 trials as training data ($\sim$ 20k frames), and 30 trials as test data ($\sim$ 6k frames). We assumed the observation follows an ARHMM, i.e. $y_t \sim \mathcal{N}(A_{z_t} y_{t-1}, \Sigma_{z_t})$. As in [126], we standardized the observations, and assumed $\{A_j, \Sigma_j\}$ follows a conjugate matrix-normal inverse-Wishart (MNIW) prior. The details of this prior are in the Appendix 4.8.4. We initialized the states $\{z_t\}_{t=1}^T$ of the three Bayesian nonparametric models using the state assignments result of a 32-states parametric ARHMM in [121].

We set $L = 40$ and ran 7 MCMC chains, each with 10000 iterations, with 9000 iterations as burn-in (Figure 4.5(c)). From Figure 4.5(a), we can see that our model has less rapid switches among states than the parametric ARHMM even if we have more states. The diagonal elements are strongly variable across states (small feature 3) (Figure 4.5(d)). For this dataset, α, ρ_1, and ρ_2 are all small, but they still have different values. Our model infers a bigger α than ρ_1, ρ_2 (big feature 1, small feature 3), as shown in Figure 4.5(e). Again, our model achieves much smaller negative predictive log-likelihood on test data than the other two Bayesian nonparametric models (Figure 4.5(b)), due to disentanglement of hyperparameters, improving the expressiveness of the HDP prior.

4.7 Discussion and conclusion

In this chapter, we propose an extension of the sticky HDP-HMM, to decouple the strength of the self-persistence prior (or the similarity of the diagonal elements of the transition matrix) and transition prior (or the similarity of the rows of the transition matrix). We develop two novel Gibbs samplers for performing efficient inference. We also show in simulated and real data that our extension outperforms existing approaches.

The work [141] proposed a Bayesian nonparametric prior to Hidden Semi-Markov Model [142] which also extends the sticky HDP-HMM. The idea is to break the Markovian assumption and instead model the distribution of duration of self-persistence for each state explicitly.

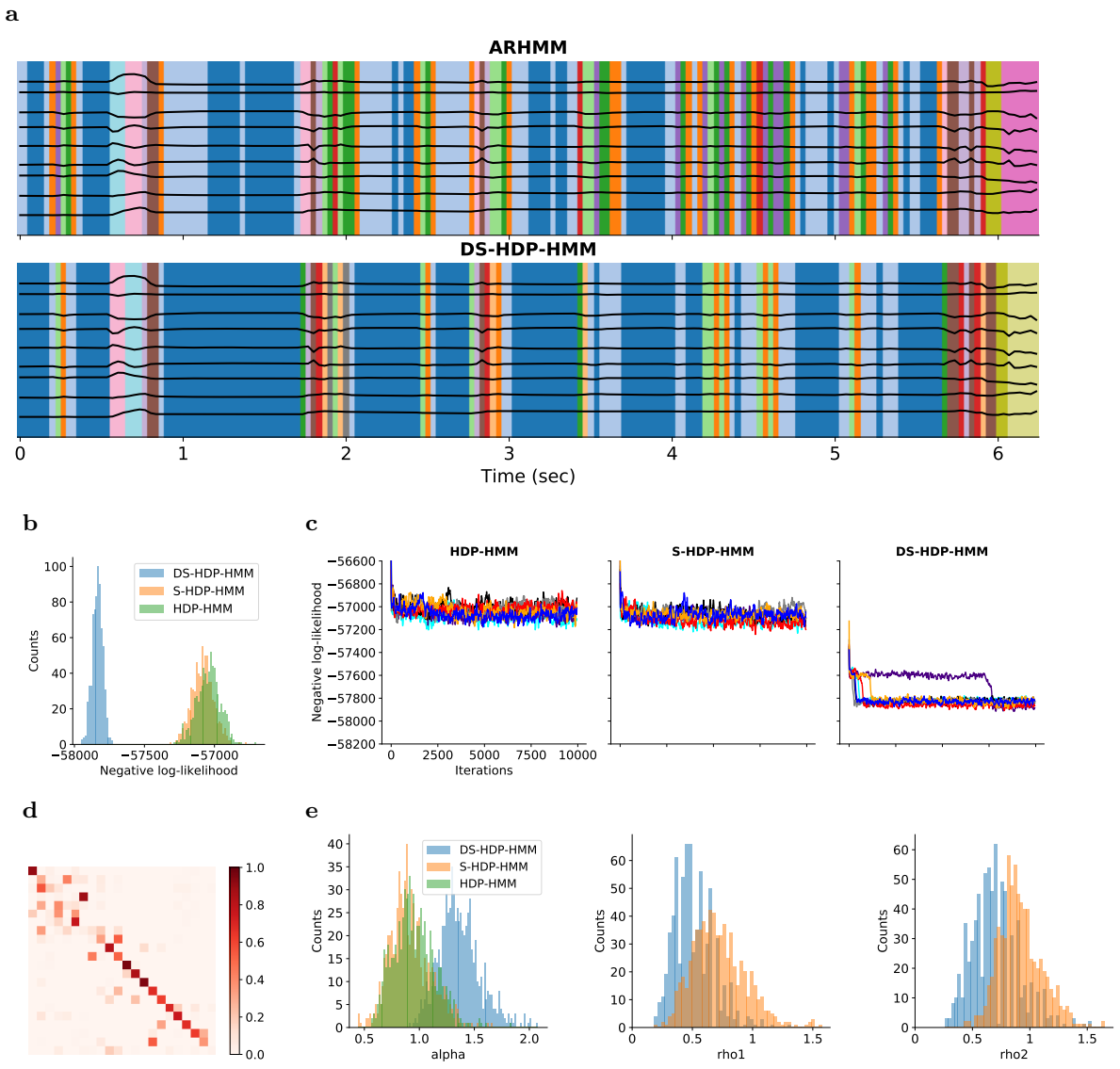

Figure 4.5: Results for mouse behavior data. (a) Observations (black) are shown on a sample trial over time, with background colors indicating the discrete state inferred for that time step using the ARHMM (number of states = 16) and our model (inferred number of states = 22) (colors are chosen to maximize the overlap between these two results). (b)(c)(e) Conventions as in the corresponding plots in Figure 4.4. Note the significant loglikelihood gap between the DS-HDP-HMM and the baseline models here. Our model infers a bigger α than ρ_1, ρ_2 (big feature 1, small feature 3). (d) Transition matrix estimated by DS-HDP-HMM. Note that the diagonal elements are strongly variable across states (small feature 3).

The flexibility of the choice of duration distribution provides a rich set of possible priors, though the choice of duration distribution can be challenging for practitioners. Our formu-

lation can be thought as a special case of their model, but with a few important advantages. First, we clarify that the limitations of the sticky HDP-HMM might not necessarily come from the Markovian assumption but could instead be due to the entanglement of the prior. Second, our extension adds a single hyperparameter, making it easy for practitioners to use. Third, our formulation still maintains the Markovian structure, which is useful in many applications. For example, there are applications of HDP-HMM prior to Markov decision processes [143, 144, 145], in which the Markovian structure is critical. Also, by harnessing the Markovian structure, our model enjoys scalable $O(TK^2)$ message passing algorithms, while the message passing algorithm of HSMM has complexity of $O(T^2K + TK^2)$ in general.

In the future we hope to explore alternative computational approaches (specifically, stochastic variational [146, 147] and amortized inference [148] methods), which may further improve the scalability of the model introduced here. Another important direction is to adapt the mixture of finite mixtures (MFM) model [149] to the HMM setting, as a replacement for the HDP prior; as in the mixture modeling setting discussed in [149], we expect that the MFM prior may lead to better estimates of the number of latent states.

Open source code is available at `https://github.com/zhd96/ds-hdp-hmm`.

4.8 Appendix

4.8.1 Notation

Table 4.1 summarizes the notation used in this chapter.

Table 4.1: Notation

General notation	
$x_{1:T}$	the sequence $\{x_1, \cdots, x_T\}$
$x_{\setminus t}$	the sequence $\{x_1, \cdots, x_{t-1}, x_{t+1}, \cdots, x_T\}$
$x_{\cdot j}$	$\sum_i x_{ij}$
$x_{i\cdot}$	$\sum_j x_{ij}$
γ	concentration parameter in DP prior
H	prior distribution of parameters
DP(γH)	Dirichlet process distribution with concentration parameter γ and base measure H
GEM(γ)	stick-breaking distribution with parameter γ
β	global transition distribution
α	concentration parameter in transition matrix prior
θ_j	parameter of distribution of observations associated with state j
π_j	transition distribution for state j
z_t	latent state at time t
y_t	observation at time t
Sticky HDP-HMM	
κ	scalar stickiness parameter
Disentangled sticky HDP-HMM	
κ_j	self-persistence probability for state j
ρ_1, ρ_2	parameters in prior distribution of self-persistence probability
$\bar{\pi}_j$	transition distribution if not self-persistent for state j
w_t	binary variable indicating next step is self-persistent or switching
n_{jk}	number of transitions from state j to k with $w_t = 0$ in $\{z_t\}_{t=1}^T$
n_{jk}^{-t}	number of transitions from state j to k with $w_t = 0$ in $\{z_t\}_{t=1}^T$, not counting z_{t-1} to z_t or z_t to z_{t+1}
m_{jk}	number of tables in restaurant j serving dish k

4.8.2 Proof of Theorem 1

Proof. The stick breaking formulation of Dirichlet process gives us the following sampling strategy of DP(β') with $\beta' = (\kappa, \alpha\beta_1, \alpha\beta_2, \cdots)$:

$$\eta_0' \sim \text{beta}(\kappa, \alpha),\ \eta_i' \sim \text{beta}(\alpha\beta_i,\ \alpha(1 - \sum_{l \le i} \beta_l)),\ i \ge 1$$

$$\eta_0 = \eta_0',\ \eta_1 = \eta_1',\ \eta_i = \eta_i' \prod_{1 \le l \le i-1} (1 - \eta_l')$$

Now we have $(\eta_0, (1-\eta_0)\eta_1, \cdots, (1-\eta_0)\eta_i, \cdots,) \sim \text{DP}(\beta')$.

As a result, $(\eta_0 + (1-\eta_0)\eta_1, (1-\eta_0)\eta_2, \cdots) \sim \mathrm{DP}(\alpha\beta + \kappa\delta_1)$. The exchangeability extends the results to general δ_j for $j \geq 1$. On the other hand, we can interpret this construction as follows. Let $\eta = (\eta_1, \cdots, \eta_k, \cdots)$, then $\eta \sim \mathrm{DP}(\alpha\beta)$. Note that η_0 is independent of η, hence this gives us $\eta_0\delta_1 + (1-\eta_0)\eta \sim \mathrm{DP}(\alpha\beta + \kappa\delta_1)$.

This exactly corresponds to our formulation with $\rho_1 = \kappa$ and $\rho_2 = \alpha$. □

4.8.3 Derivation of Gibbs samplers

Derivation of direct assignment sampler

Step 1: sequentially sample z_t, w_t, w_{t+1} The joint posterior of z_t, w_t, and w_{t+1} has the following form

$$\begin{aligned} & p(z_t = k, w_t, w_{t+1} | z_{\setminus t}, w_{\setminus\{t,t+1\}}, y_{1:T}, \alpha, \beta, \{\kappa_j\}_{j=1}^{K+1}) \\ \propto\ & p(z_t = k, w_t, w_{t+1} | z_{\setminus t}, w_{\setminus\{t,t+1\}}, \alpha, \beta, \{\kappa_j\}_{j=1}^{K+1}) \cdot p(y_t | y_{\setminus t}, z_t = k, z_{\setminus t}). \end{aligned}$$

where $y_{\setminus t}$ are all the observations except for y_t, $w_{\setminus\{t,t+1\}}$ are all the w_t except for w_t, w_{t+1}.

The predictive observation likelihood $p(y_t | y_{\setminus t}, z_t = k, z_{\setminus t})$ can be easily computed if we use a conjugate prior on the parameter in observation likelihood. For all the emissions in the main chapter, we used conjugate priors. See the normal-inverse-Wishart conjugate model in [150], Dirichlet-multinomial conjugate model in [151], Poisson-gamma conjugate model in [138], and matrix-normal inverse-Wishart conjugate model in [126].

$$
\begin{aligned}
& p(z_t = k, w_t, w_{t+1} | z_{\setminus t}, w_{\setminus\{t,t+1\}}, \alpha, \beta, \{\kappa_j\}_{j=1}^{K+1}) \\
\propto\ & p(z_t = k, w_t, w_{t+1}, z_{t+1} | z_{\setminus\{t,t+1\}}, w_{\setminus\{t,t+1\}}, \alpha, \beta, \{\kappa_j\}_{j=1}^{K+1}) \\
\propto\ & \int_{\boldsymbol{\pi}} p(w_t|\kappa_{z_{t-1}}) p(z_t|w_t, \pi_{z_{t-1}}) p(w_{t+1}|\kappa_{z_t}) p(z_{t+1}|w_{t+1}, \pi_{z_t}) \\
& \prod_i (p(\pi_i|\alpha,\beta) \prod_{\tau|z_{\tau-1}=i, w_\tau=0, \tau\neq t,t+1} p(z_\tau|\pi_i)) d\boldsymbol{\pi} \\
\propto\ & \int_{\boldsymbol{\pi}} p(w_t|\kappa_{z_{t-1}}) p(z_t|w_t, \pi_{z_{t-1}}) p(w_{t+1}|\kappa_{z_t}) p(z_{t+1}|w_{t+1}, \pi_{z_t}) \\
& \prod_i p(\pi_i | \{\tau | z_{\tau-1} = i, w_\tau = 0, \tau \neq t, t+1\}, \alpha, \beta) d\boldsymbol{\pi}.
\end{aligned}
\tag{4.8}
$$

Let $z_{t-1} = j, z_{t+1} = l$, then equation 4.8 has the following cases

$$
\begin{cases}
\kappa_j^2, & \text{if } w_t = w_{t+1} = 1, k = j = l \\
(1-\kappa_j)\kappa_l \int_{\pi_j} p(z_t = l|w_t, \pi_j) p(\pi_j | \{\tau | z_{\tau-1} = j, w_\tau = 0, \tau \neq t\}, \alpha, \beta) d\pi_j, & \text{if } w_t = 0, w_{t+1} = 1, k = l \\
(1-\kappa_j)\kappa_j \int_{\pi_j} p(z_{t+1} = l|w_{t+1}, \pi_j) p(\pi_j | \{\tau | z_{\tau-1} = j, w_\tau = 0, \tau \neq t+1\}, \alpha, \beta) d\pi_j, & \text{if } w_t = 1, w_{t+1} = 0, k = j \\
(1-\kappa_j)(1-\kappa_k) \int_{\pi_j} p(z_t = k|w_t, \pi_j) p(\pi_j | \{\tau | z_{\tau-1} = j, w_\tau = 0, \tau \neq t\}, \alpha, \beta) d\pi_j \cdot & \\
\int_{\pi_k} p(z_{t+1} = l|w_{t+1}, \pi_k) p(\pi_k | \{\tau | z_{\tau-1} = k, w_\tau = 0, \tau \neq t+1\}, \alpha, \beta) d\pi_k, & \text{if } w_t = 0, w_{t+1} = 0, k \neq j \\
(1-\kappa_j)(1-\kappa_k) \cdot & \\
\int_{\pi_j} p(z_t = j|w_t, \pi_j) p(z_{t+1} = l|w_{t+1}, \pi_j) p(\pi_j | \{\tau | z_{\tau-1} = j, w_\tau = 0, \tau \neq t, t+1\}, \alpha, \beta) d\pi_j, & \text{if } w_t = 0, w_{t+1} = 0, k = j \\
0, & \text{otherwise}
\end{cases}
$$

Similar as [122], we have the following equations with a close-form integration.

$$\int_{\pi_j} p(z_t = k|w_t, \pi_j)p(\pi_j|\{\tau|z_{\tau-1} = j, w_\tau = 0, \tau \neq t\}, \alpha, \beta)d\pi_j = \frac{\alpha\beta_k + n_{jk}^{-t}}{\alpha + n_{j\cdot}^{-t}},$$

$$\int_{\pi_k} p(z_{t+1} = l|w_{t+1}, \pi_k)p(\pi_k|\{\tau|z_{\tau-1} = k, w_\tau = 0, \tau \neq t+1\}, \alpha, \beta)d\pi_k = \frac{\alpha\beta_l + n_{kl}^{-t}}{\alpha + n_{k\cdot}^{-t}},$$

$$\int_{\pi_j} p(z_t = j|w_t, \pi_j)p(z_{t+1} = l|w_{t+1}, \pi_j)p(\pi_j|\{\tau|z_{\tau-1} = j, w_\tau = 0, \tau \neq t, t+1\}, \alpha, \beta)d\pi_j$$

$$= \frac{(\alpha\beta_j + n_{jj}^{-t})(\alpha\beta_l + n_{jl}^{-t} + \delta(j,l))}{(\alpha + n_{j\cdot}^{-t})(\alpha + n_{j\cdot}^{-t} + 1)},$$

$$\int_{\pi_j} p(z_t = k|w_t, \pi_j)p(\pi_j|\{\tau|z_{\tau-1} = j, w_\tau = 0, \tau \neq t\}, \alpha, \beta)d\pi_j \cdot$$

$$\int_{\pi_k} p(z_{t+1} = l|w_{t+1}, \pi_k)p(\pi_k|\{\tau|z_{\tau-1} = k, w_\tau = 0, \tau \neq t+1\}, \alpha, \beta)d\pi_k$$

$$= \frac{\alpha\beta_k + n_{jk}^{-t}}{\alpha + n_{j\cdot}^{-t}} \frac{\alpha\beta_l + n_{kl}^{-t}}{\alpha + n_{k\cdot}^{-t}},$$

where $n_{z_{t-1}z_t}$ is the number of transitions from state z_{t-1} to state z_t with $w_t = 0$ in $z_{1:T}$. $n_{z_{t-1}z_t}^{-t}$ is the number of transitions from state z_{t-1} to z_t with $w_t = 0$, without counting the transitions z_{t-1} to z_t or z_t to z_{t+1}. $n_{j\cdot}$ is the sum of number of transitions from j to other states.

If $z_t = K + 1$, i.e. a new state appears, we will increment K, sample self-persistence probability κ_{K+1} for a new state from prior, and update β in the following stick-breaking way. Sample $b \sim \text{beta}(1, \gamma)$, and assign $\beta_K = b\beta_{k_{\text{new}}}, \beta_{k_{\text{new}}} = (1-b)\beta_{k_{\text{new}}}$, where $\beta_{k_{\text{new}}} = \sum_{i=K+1}^{\infty} \beta_i$.

Step 2: sample $\{\kappa_j\}_{j=1}^{K+1}$ The posterior of κ_j is derived as follows using the beta-binomial conjugate property

$$\kappa_j \sim \text{beta}(\rho_1 + \sum_{\tau, z_{\tau-1}=j} w_\tau, \rho_2 + \sum_{\tau, z_{\tau-1}=j} 1 - w_\tau),\ j = 1, \cdots, K+1,$$

where κ_{K+1} serves for the self-persistence probability of a new state.

Step 3: sample β To sample the global transition distribution β, we introduce auxiliary variables m_{jk} is the auxiliary variable according to the Chinese restaurant franchise (CRF) formulation of HDP prior [68]. (Note that it can be thought of as the number of tables in restaurant j which serve dish k.) We first update m_{jk}, then sample β.

For each $(j,k) \in \{1,\cdots,K\}^2$, set $m_{jk} = 0, s = 0$. Then for $i = 1,\cdots,n_{jk}$, sample $x \sim \text{Ber}(\frac{\alpha\beta_k}{n+\alpha\beta_k})$. Increment s, and if $x = 1$ increment m_{jk}. Now we can sample β as

$$(\beta_1, \beta_2, \cdots, \beta_K, \beta_{k_{\text{new}}}) \sim \text{Dir}(m_{\cdot 1}, \cdots, m_{\cdot K}, \gamma),$$

where $\beta_{k_{\text{new}}} = \sum_{i=K+1}^{\infty} \beta_i$ is for transiting to a new state, $m_{\cdot k}$ is $\sum_{j=1}^{K} m_{jk}$. A more detailed derivation of the sampling of β can be found in [122].

Step 4: sample hyperparameters $\alpha, \gamma, \rho_1, \rho_2$ Sampling α, γ are the same as in [68, 152]. Basically, if we apply gamma prior on α and γ, then by introducing some auxiliary variables, we can have the posterior of α and γ to be gamma-conjugate.

For ρ_1, ρ_2, we follow the reparametrization trick discussed in Chapter 5 [153]. Basically, we reparametrize $\phi = \frac{\rho_1}{\rho_1+\rho_2}, \eta = (\rho_1+\rho_2)^{-1/3}$, and apply a Uniform$([0,1]\times[0,2])$ prior on (ϕ,η). Then we can discretize the support of (ϕ,η) and numerically compute the posterior. An alternative way to avoid the discretization would be to use Metropolis-Hasting sampling [77].

Derivation of weak-limit sampler

Step 1: jointly sample $\{z_t, w_t\}_{t=1}^T$ The joint conditional distribution of $z_{1:T}, w_{1:T}$ is

$$\begin{aligned} p(z_{1:T}, w_{1:T}|y_{1:T}, \bar{\pi}, \{\kappa_j\}_{j=1}^L, \theta) = & \, p(z_T, w_T|z_{T-1}, y_{1:T}, \bar{\pi}, \{\kappa_j\}_{j=1}^L, \theta) \\ & p(z_{T-1}, w_{T-1}|z_{T-2}, y_{1:T}, \bar{\pi}, \{\kappa_j\}_{j=1}^L, \theta) \\ & \cdots p(z_1|y_{1:T}, \bar{\pi}, \{\kappa_j\}_{j=1}^L, \theta). \end{aligned}$$

The conditional distribution of z_1 is:

$$p(z_1|y_{1:T}, \bar{\pi}, \{\kappa_j\}_{j=1}^L, \theta) \propto \; p(z_1)p(y_1|\theta_{z_1})p(y_{2:T}|z_1, \bar{\pi}, \{\kappa_j\}_{j=1}^L, \theta)$$

The conditional distribution of $z_t, t > 1$ is:

$$\begin{aligned}
& p(z_t, w_t|z_{t-1}, y_{1:T}, \bar{\pi}, \{\kappa_j\}_{j=1}^L, \theta) \\
\propto\; & p(z_t, w_t, y_{1:T}|z_{t-1}, \bar{\pi}, \{\kappa_j\}_{j=1}^L, \theta) \\
=\; & p(z_t|\bar{\pi}_{z_{t-1}}, w_t)p(w_t|z_{t-1}, \{\kappa_j\}_{j=1}^L)p(y_{t:T}|z_t, \bar{\pi}, \{\kappa_j\}_{j=1}^L, \theta)p(y_{1:t-1}|z_{t-1}, \bar{\pi}, \{\kappa_j\}_{j=1}^L, \theta) \\
\propto\; & p(z_t|\bar{\pi}_{z_{t-1}}, w_t)p(w_t|z_{t-1}, \{\kappa_j\}_{j=1}^L)p(y_{t:T}|z_t, \bar{\pi}, \{\kappa_j\}_{j=1}^L, \theta) \\
=\; & p(z_t|\bar{\pi}_{z_{t-1}}, w_t)p(w_t|z_{t-1}, \{\kappa_j\}_{j=1}^L)p(y_t|\theta_{z_t})m_{t+1,t}(z_t),
\end{aligned}$$

where $m_{t+1,t}(z_t) \triangleq p(y_{t+1:T}|z_t, \bar{\pi}, \{\kappa_j\}_{j=1}^L, \theta)$, which is the backward message passed from z_t to z_{t-1} and for an HMM is recursively defined by:

$$\begin{aligned}
& m_{t+1,t}(z_t) = \sum_{z_{t+1}, w_{t+1}} p(z_{t+1}|\bar{\pi}_{z_t}, w_{t+1})p(w_{t+1}|z_t, \{\kappa_j\}_{j=1}^L)p(y_{t+1}|\theta_{z_{t+1}})m_{t+2,t+1}(z_{t+1}), t \leq T \\
& m_{T+1,T}(z_T) = 1
\end{aligned}$$

Step 2: sample $\{\kappa_j\}_{j=1}^L$ Same as 4.8.3 step 2, but for $j = 1, \cdots, L$.

Step 3: sample $\{\beta_j\}_{j=1}^L, \{\bar{\pi}_j\}_{j=1}^L$

$$\begin{aligned}
\beta|m, \gamma &\sim \text{Dir}\left(\gamma/L + m_{\cdot 1}, \cdots, \gamma/L + m_{\cdot L}\right), \\
\bar{\pi}_j|z_{1:T}, w_{1:T}, \alpha, \beta &\sim \text{Dir}(\alpha\beta_1 + n_{j1}, \cdots, \alpha\beta_L + n_{jL}), \; j = 1, \cdots, L.
\end{aligned}$$

Step 4: sample $\{\theta_j\}_{j=1}^L$ We sample θ from the posterior distribution depending on the emission function and the base measure H of the parameter space Θ, i.e. $\theta_j|z_{1:T}, y_{1:T} \sim p(\theta_j|\{y_t|z_t = j\})$. For all the emissions in the main chapter, we can easily sample θ_j from its

posterior using the conjugacy properties.

Step 5: sample hyperparameters $\alpha, \gamma, \rho_1, \rho_2$ Same as 4.8.3 step 4.

4.8.4 Details of prior in ARHMM

We used the MNIW prior in ARHMM, which is given by placing the matrix-normal prior: $\mathcal{MN}(M, \Sigma_j, V)$ on A_j given Σ_j:

$$p(A_j|\Sigma_j) = \frac{1}{(2\pi)^{d^2/2}|V|^{d/2}|\Sigma_j|^{d/2}} \exp\left(-\frac{1}{2} tr\left[(A_j - M)^\top \Sigma_j^{-1} (A_j - M) V^{-1}\right]\right),$$

where M is $d \times d$ matrix, Σ_j, V are $d \times d$ positive-definite matrix, d is the dimension of observation y_t; and an inverse-Wishart prior $IW(S_0, n_0)$ on Σ_j:

$$p(\Sigma_j) = \frac{|S_0|^{n_0/2}}{2^{n_0 d/2}\Gamma_d(n_0/2)} |\Sigma_j|^{-(n_0+d+1)/2} \exp\left(-\frac{1}{2} tr\left(\Sigma_j^{-1} S_0\right)\right),$$

where $\Gamma_d(\cdot)$ is the multivariate gamma function.

We set $M = \mathbf{0}, V = I_{d\times d}, n_0 = d + 2, S_0 = 0.75\bar{\Sigma}$, where $\bar{\Sigma} = \frac{1}{T}\sum_{t=1}^{T}(y_t - \bar{y})(y_t - \bar{y})^\top$ when segmenting the mouse behavior video.

4.8.5 Simulation results for Gaussian emission

We simulated data using a transition matrix (Figure 4.6(a)) with different κ_j and the same $\bar{\pi}_j$ across states, which corresponds to a large transition concentration parameter α (big feature 1), and small self-persistence parameters ρ_1, ρ_2 (small feature 3). To generate Gaussian observations, we assumed that $y_t \sim \mathcal{N}(\theta_{z_t}, 0.5^2), \theta_j \overset{iid}{\sim} \mathcal{N}(3.5, 6^2),\ j = 1, \cdots, 8$. We placed a Gaussian prior on the Gaussian mean parameters, with mean and variance equal to the empirical mean and variance, and fixed the observation noise as $\mathcal{N}(0, 0.5^2)$. We ran 3 MCMC chains, each with 30000 iterations, with 26000 iterations as burn-in. See Figure 4.6 for the results.

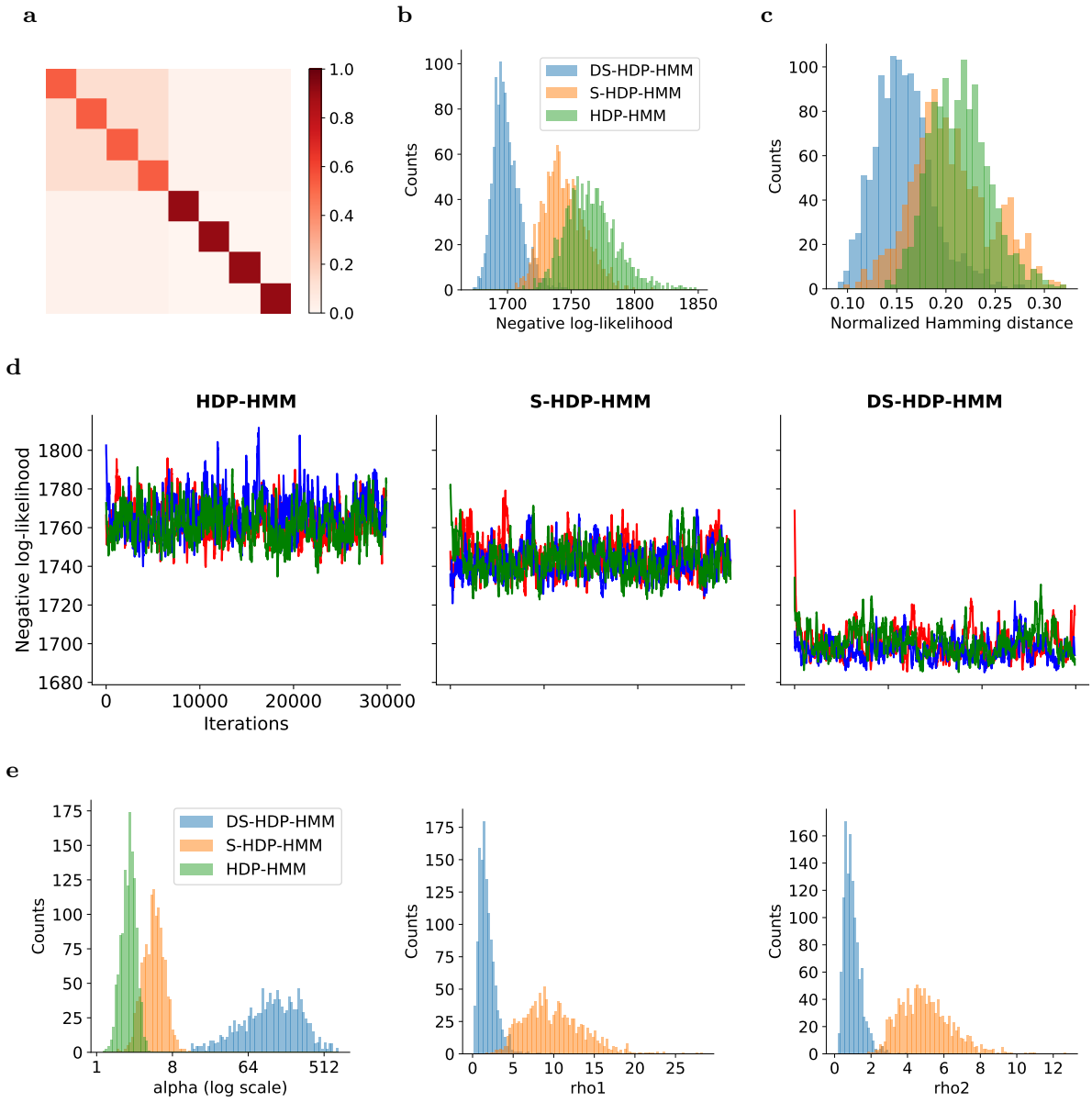

Figure 4.6: Results for simulated data with different self-persistence (ρ_1, ρ_2 small), same transition (α large) and Gaussian emission. We ran 3 chains, each with 30000 iterations, with 26000 iterations as burn-in for this data. Conventions as in Figure 4.2 in the main chapter. Our model learns a big α and small ρ_1, ρ_2, which is consistent with the data.

Chapter 5: Learning identifiable and interpretable latent models of high-dimensional neural activity using pi-VAE

This chapter was published as "Learning identifiable and interpretable latent models of high-dimensional neural activity using pi-VAE" [86] in Advances in Neural Information Processing Systems 33 (NeurIPS 2020) with Xue-Xin Wei. We thank Matthew G. Perich and Lee E. Miller for sharing the monkey reaching data, Kenneth Kay, Yuanjun Gao, and Liam Paninski for helpful comments and discussions, as well as three anonymous reviewers for their feedback which helped improving the paper.

5.1 Introduction

Popular analysis methods of neural responses in neurophysiology are mainly in two classes: one based on regression, the other on latent variable modeling. Generalized Linear Model (GLM, e.g., [154, 155]) and tuning curve analysis [156, 157, 158] are notable examples of the regression-based approach, and both have been widely used in neuroscience in the past few decades [159, 160, 157, 161, 162]. These methods express the neural firing rate as a function of the stimulus variable, thus naturally define encoding models, which can be inverted to decode the stimulus variables [156, 163, 157, 164, 165]. In contrast, the latent-based approach aims to account to variability of the neural responses using a relatively small number of latent variables which are typically not observed. Recently, various latent-based methods have been developed or applied to analyze neural data and in particular simultaneously recorded neural population data, including principal component analysis [166, 167, 168, 169, 170], factor analysis [171, 172, 173], linear/nonlinear dynamical systems [174, 175, 71, 176, 72], among others (e.g., [177, 178, 179, 180]). Each class of models carries certain

advantages and disadvantages. Regression-based methods tend to have higher interpretability, however, they often suffer the problem of under-fitting. Latent-based models are more flexible in accounting for the neural variability, however they may be difficult to interpret and sometimes not identifiable. Notably, some studies had incorporated latent fluctuations into the encoding models [181, 182, 112, 183, 184, 162] yielding promising results, although these models often assumed highly specialized latent structure, thus potentially limits the applicability in practice.

The issues of identifiability and interpretability are becoming increasingly important as the neuroscience community adapts more sophisticated methods from nonlinear deep generative models [176, 185]. Deep generative models have the promise of extracting complex nonlinear structure which may be difficult to achieve by linear methods, as demonstrated by recent work based on the variational auto-encoder (VAE) (e.g., [46, 186, 71, 72]). However, over the past few years it has become increasingly clear that the latents extracted from these models, and VAE in particular, are often highly entangled therefore difficult to interpret [187, 50, 188, 51, 48, 52, 49]. Given these considerations, an important question is how to model neural population responses with nonlinear models that are powerful yet scientifically insightful via identifiability and interpretability.

We propose a model formulation which represents one step toward addressing this question. Specifically, we draw on recent progress on identifiable VAE (iVAE) [49, 53], and generalize and adapt it to make it directly applicable to a broad variety of datasets. Conceptually, our method combines the respective strengths of regression-based and latent-based approaches (Figure 5.1): i) by using the VAE architecture, it is expressive and flexible; ii) by treating task variables as labels and explicitly modeling them with the latents, our method is better constrained and under certain conditions identifiable. We apply our method to synthetic data and eletrophysiological datasets from population recording of rat hippocampus in a navigation task [189, 190] and the motor area of macaque during a reaching task [191]. We demonstrate that our method can recover interpretable latent structure that is informative

about the structure of the neural code and dynamics.

5.2 Model

Notations We denote $\mathbf{x} \in \mathbb{R}^n$ as observations. For our purpose, $\mathbf{x}$ specifically represents the population response or spike counts within a small time window. We use $\mathbf{u} \in \mathbb{R}^d$ to represent *task variables* (or *labels*), that are measured along with the neural activities, e.g., the location of the animal when studying navigation tasks. $\mathbf{u}$ can be discrete or continuous. Additionally, we denote $\mathbf{z} \in \mathbb{R}^m (m \ll n)$ as the unobserved low-dimensional latent variables.

5.2.1 Generative model

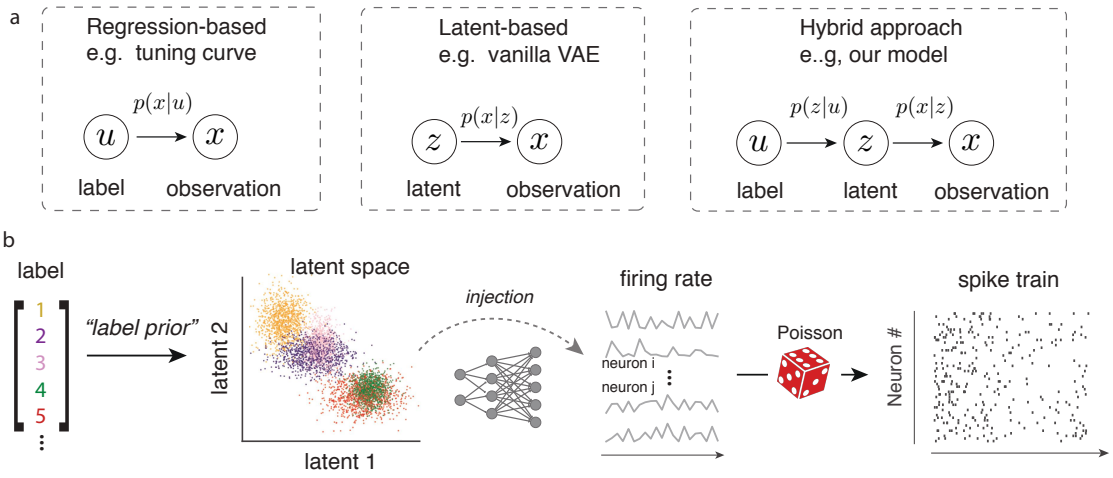

Figure 5.1: The model framework and generative model. (a) Structure of three classes of statistical models for neural data analysis. Our method is based on the integration of the first two classes into a hybrid approach. Our approach models the statistical dependence between label ($\mathbf{u}$) and latent ($\mathbf{z}$) as well as between latent ($\mathbf{z}$) and observation ($\mathbf{x}$) simultaneously. (b) Schematic illustration of the generative model of pi-VAE. Major components include the "label prior" between the task variables and the latent, an injective mapping between latent and firing rate parameterized by normalizing flow, and Poisson observation noise.

Our goal is to develop models that are flexible and expressive in capturing the variability of the data, while also well-constrained so that the models would enjoy identifiability and interpretability. Motivated by these considerations, we propose a generative model formulation which integrates key ingredients of the latent-based and regression-based approaches

(see Figure 5.1a):

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}|\mathbf{u}) = p_{\mathbf{f}}(\mathbf{x}|\mathbf{z}) p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}). \tag{5.1}$$

This is a general formulation, and some previous models may be re-formulated to conform with it (e.g., [182, 112, 183]). In this chapter, we will focus on a specific implementation that is directly inspired by the recent work on identifiable VAE [49, 192, 53]. In the interest of neuroscience applications, we have developed a method that can simultaneously deal with Poisson noise, both discrete and continuous labels, and larger output dimension than input dimension (Figure 5.1b). We will show that our model is sufficiently expressive for many applications, yet still constrained enough to be identifiable.

We start by defining the component that describes the relation between the label and the latent, *i.e.*, $p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u})$. We will refer to it as the *"label prior"*. Following [49], we assume $p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u})$ to be conditionally independent, where each element $\mathbf{z}_i \in \mathbf{z}$ has *an exponential family distribution* given $\mathbf{u}$,

$$p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}) = \prod_{i=1}^{m} p(\mathbf{z}_i|\mathbf{u}) = \prod_{i=1}^{m} \frac{Q_i(\mathbf{z}_i)}{Z_i(\mathbf{u})} \exp\left[\sum_{j=1}^{k} T_{i,j}(\mathbf{z}_i)\lambda_{i,j}(\mathbf{u})\right], \tag{5.2}$$

where Q_i is the base measure, $\mathbf{T}_i = (T_{i,1}, \cdots, T_{i,k})$ are the sufficient statistics, $Z_i(\mathbf{u})$ is the normalizing factor, $\boldsymbol{\lambda}_i = (\lambda_{i,1}(\mathbf{u}), \cdots, \lambda_{i,k}(\mathbf{u}))$ are the natural parameters, and k is pre-defined number of sufficient statistics. Practically, a small number of components k is often sufficient for the problems which we have considered. For discrete $\mathbf{u}$, we simply use a different λ_{ij} for different $\mathbf{u}$. To deal with continuous $\mathbf{u}$, we develop a procedure by parameterizing λ_{ij} as a function of $\mathbf{u}$ using a feed-forward neural network. Details are given in the Appendix.

We next turn to the dependence between the latent and observation, *i.e.*, $p_{\mathbf{f}}(\mathbf{x}|\mathbf{z})$. In [49], $p_{\mathbf{f}}(\mathbf{x}|\mathbf{z})$ is defined using additive noise $p_{\mathbf{f}}(\mathbf{x}|\mathbf{z}) = p_{\boldsymbol{\epsilon}}(\mathbf{x} - \mathbf{f}(\mathbf{z}))$, i.e. $\mathbf{x} = \mathbf{f}(\mathbf{z}) + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is an independent noise variable. To model the spike data, we generalize it to Poisson model $p_{\mathbf{f}}(\mathbf{x}|\mathbf{z}) = \text{Poisson}(\mathbf{f}(\mathbf{z}))$ with $\mathbf{f}$ being the instantaneous firing rate to deal with the count observations. We implement $\mathbf{f}$ using normalizing flow as detailed later. Putting together,

we denote $\boldsymbol{\theta} = (\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda})$ as parameters in generative model 5.1. We refer to our model as Poission identifiable VAE, or pi-VAE for simplicity.

Identifiability [49] has proved that the additive noise model is identifiable with certain assumptions. Under the same assumptions, we can prove that pi-VAE is also identifiable.

Definition 1. *Let $\sim$ be an equivalence relation defined on the domain of parameters: $\Theta = (\boldsymbol{\theta} := (\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda}))$. Model 5.1 is said to be identifiable up to $\sim$ if $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{u}) = p_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}|\mathbf{u}) \implies \boldsymbol{\theta} \sim \tilde{\boldsymbol{\theta}}$.*

Definition 2. *Define $\sim$ as $\boldsymbol{\theta} \sim \tilde{\boldsymbol{\theta}} \iff \exists A, c, \mathbf{T}\left(\mathbf{f}^{-1}(\mathbf{x})\right) = A\tilde{\mathbf{T}}\left(\tilde{\mathbf{f}}^{-1}(\mathbf{x})\right) + c, \forall \mathbf{x} \in \mathrm{Img}(\mathbf{f}) \subset \mathbb{R}^n$, where $\boldsymbol{\theta} = (\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda}), \tilde{\boldsymbol{\theta}} = (\tilde{\mathbf{f}}, \tilde{\mathbf{T}}, \tilde{\boldsymbol{\lambda}})$, A is a full rank $mk \times mk$ matrix, $c \in \mathbb{R}^{mk}$ is a vector.*

Theorem 2. *Assume that we observe data sampled from pi-VAE model defined according to equation 5.1,5.2 with Poisson noise and parameters $\boldsymbol{\theta} = (\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda})$. Assume the following holds:*

i) The firing rate function $\mathbf{f}$ in equation 5.1 is injective,

ii) The sufficient statistics $T_{i,j}$ in 5.2 are differentiable almost everywhere, and their derivatives $T'_{i,j}$ are nonzero almost everywhere for $1 \leq i \leq m, 1 \leq j \leq k$,

iii) There exists $mk + 1$ distinct points $\mathbf{u}^0, \cdots, \mathbf{u}^{mk}$ such that the matrix

$$L = \left(\boldsymbol{\lambda}(\mathbf{u}^1) - \boldsymbol{\lambda}(\mathbf{u}^0), \cdots, \boldsymbol{\lambda}(\mathbf{u}^{mk}) - \boldsymbol{\lambda}(\mathbf{u}^0)\right)$$

of size $mk \times mk$ is invertible, then the pi-VAE model is identifiable up to $\sim$.

This theorem is a straight-forward generalization of the results in [49]. Proof is given in the Appendix. This theorem means, if two sets of model parameters lead to the same marginal distribution of $\mathbf{x}$, then $(\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda}) \sim (\tilde{\mathbf{f}}, \tilde{\mathbf{T}}, \tilde{\boldsymbol{\lambda}})$. Thus one can hope to recover posterior distribution $p(\mathbf{z}|\mathbf{x})$ up to a linear transformation A and point-wise non-linearities between $\mathbf{T}$ and $\tilde{\mathbf{T}}$, as well as the joint distribution $p(\mathbf{x}, \mathbf{z})$. Note that other forms of identifiability may be derived with modified assumptions [49]. While practically, without knowing the

ground truth, the assumptions may be difficult to verify, some encouraging preliminary evidence suggest that identifiability may have some robustness with mild violations of model assumptions [53].

We next describe how to parameterize the injection $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^n$. Here we extend the General Incompressible-flow Network (GIN) proposed in [53], which shares the flexibility of RealNVP [193] and the volume-preserving property of NICE [194]. Practically, we found our implementation to be reasonably efficient computationally. Specifically, GIN defines a mapping from $\mathbb{R}^D \to \mathbb{R}^D$ with Jacobian determinant equal to 1 [53]. It splits the D-dimensional input $\mathbf{x}$ into two parts $\mathbf{x}_{1:l}, \mathbf{x}_{l+1:D}$, where $l < D$. The output $\mathbf{y}$ is defined as the concatenation of $\mathbf{y}_{1:l}$ and $\mathbf{y}_{l+1:D}$,

$$\mathbf{y}_{1:l} = \mathbf{x}_{1:l} \tag{5.3}$$

$$\mathbf{y}_{l+1:D} = \mathbf{x}_{l+1:D} \odot \exp\left(s\left(\mathbf{x}_{1:l}\right)\right) + t\left(\mathbf{x}_{1:l}\right), \tag{5.4}$$

where $s(\cdot)$ and $t(\cdot)$ are both functions defined on $\mathbb{R}^l \to \mathbb{R}^{D-l}$, and the total sum of $s\left(\mathbf{x}_{1:l}\right)$ is constrained to be zero by setting the final component to be the negative sum of previous components.

The original GIN [53] only deals with the case where the input and output have the same dimensions. In our case, the output dimension is often much larger than the input dimension. We thus develop a new scheme to parameterize $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^n$ which retains the properties of GIN. We first map $\mathbf{z}_{1:m}$ to the concatenation of $\mathbf{z}_{1:m}$ and $t\left(\mathbf{z}_{1:m}\right)$. Note that this is equivalent to GIN model with input as $\mathbf{z}_{1:m}$ padding n-m zeros. We then use several GIN blocks to map from $\mathbb{R}^n \to \mathbb{R}^n$. Recalling that each GIN block is an injection (since part of it is an identity map, one can not map two different inputs to the same output), it follows that the composition of several blocks remains an injection. While we use an extension of GIN [53] to implement the injection f here, conceivably other implementations should be possible, e.g., multi-layer perceptron with increasing number of nodes from earlier to later

layers. The efficiency of the different implementations will need to be evaluated in future.

5.2.2 Inference algorithm

The inference procedure is a modification of VAE [46]. Our algorithm simultaneously learns the deep generative model and the approximate posterior $q(\mathbf{z}|\mathbf{x},\mathbf{u})$ of true posterior $p(\mathbf{z}|\mathbf{x},\mathbf{u})$ by maximizing $\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi})$, which is the evidence lower bound (ELBO) of $p(\mathbf{x}|\mathbf{u})$,

$$\log p(\mathbf{x}|\mathbf{u}) \geq \mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi}) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{u})}\left[\log\left(p(\mathbf{x},\mathbf{z}|\mathbf{u})\right) - \log\left(q(\mathbf{z}|\mathbf{x},\mathbf{u})\right)\right]. \tag{5.5}$$

Similar to [195], we decompose the approximate posterior as

$$q(\mathbf{z}|\mathbf{x},\mathbf{u}) \propto q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}), \tag{5.6}$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is assumed to be conditionally independent exponential family distribution, i.e. $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \prod_{i=1}^{m} q(\mathbf{z}_i|\mathbf{x})$, and is parameterized by neural network. $p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u})$ is defined in equation 5.2.

We modeled both $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}), p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u})$ as independent Gaussian distribution, used the same network architecture (see Appendix for details) as well as Adam optimizer [120] with learning rate equal to 5×10^{-4}, and other values were set to the recommendation values for all the experiments in this chapter.

Inferring the latent After learning $q(\mathbf{z}|\mathbf{x},\mathbf{u})$, the latent from pi-VAE model can be inferred by computing the posterior mean. It is also of interest to infer the latent without using the label prior $p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u})$, which could be done by computing the posterior mean of $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ instead.

Decoding the label Because pi-VAE defines an encoding model on the label, one can examine how well the label could be decoded from the neural activity, which also provides

a way to check the validity of the model. Under our model formulation, decoding could be done by Bayesian rule and Monte Carlo sampling: $p(\mathbf{u}|\mathbf{x}) \propto \int p(\mathbf{x}|\mathbf{z},\mathbf{u})p(\mathbf{z}|\mathbf{u})p(\mathbf{u})d\mathbf{z}$, where the integration on right hand side can be computed through randomly sampling $p(\mathbf{z}|\mathbf{u})$. We assume a uniform prior on $\mathbf{u}$.

5.3 Results

5.3.1 Validation using synthetic data

We validated pi-VAE using synthetic data generated from models with continuous or discrete labels.

Discrete label We generated 2-dimensional latent samples $\mathbf{z}$ from a five clusters Gaussian mixture model, similar to [49] (see Figure 5.2a). The mean of each cluster was chosen independently from a uniform distribution on $[-5, 5]$ and variance from a uniform distribution on $[0.5, 3]$. These latent samples were then mapped to the mean firing rate of 100-dimensional Poisson observations through a RealNVP network (details in Appendix). Example results are shown in Figure 5.2a-d.

Continuous label We generated $\mathbf{u}$ from a uniform distribution on $[0, 2\pi]$, and latent samples $\mathbf{z}$ as a 2-dimensional independent Gaussian distribution with mean being $(\mathbf{u}, 2\sin\mathbf{u})$, and variance being $(0.6 - 0.3|\sin\mathbf{u}|, 0.3|\sin\mathbf{u}|)$. Observations were generated in the same way as simulation of discrete label. Example results are shown in Figure 5.2e-h.

Based on these and other numerical experiments, we found that in general pi-VAE could reliably uncover latent structure similar to ground truth for both discrete and continuous labels, while VAE often leads to more distorted latent. Note that our VAE implementation is similar to pi-VAE except that no label prior is used (e.g., Poisson observation noise is still assumed). We also found pi-VAE without label prior (during the inference) still led to reasonably good recovery of the latent (Figure 5.2c,g), suggesting that incorporating label

prior could help with learning a better model, not just inference.

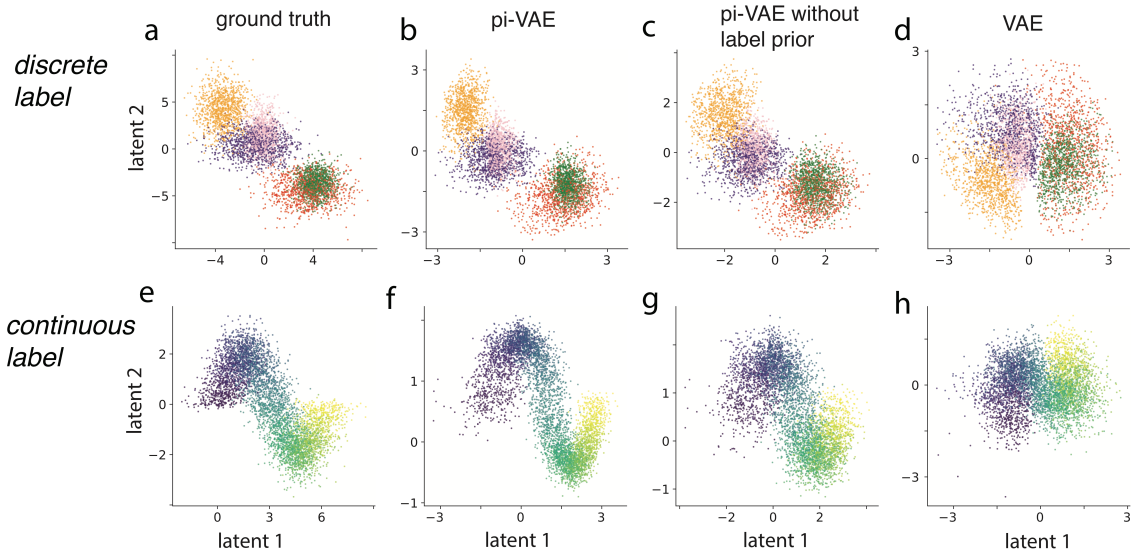

Figure 5.2: Example numerical experiments, suggesting that pi-VAE, but not VAE, could identify latent structure. (a) True latent variables, simulated based on discrete label, (b) mean of the latent posterior $q(\mathbf{z}|\mathbf{x},\mathbf{u})$ estimated from pi-VAE, (c) mean of $q(\mathbf{z}|\mathbf{x})$ from pi-VAE, (d) mean of the latent posterior from VAE, that is, the Bayesian estimate *inferred* without the label prior. (e-h) similar to (a-d), but for a simulation based on continuous label.

5.3.2 Applications to neural population data

We have applied pi-VAE to analyze two electrophysiology datasets, each has more than 100 simultaneously recorded neurons when the animals were performing behavioral tasks. In these real data applications the ground truth is unknown and the assumptions required by identifiability may be violated [53], making it difficult to assess identifiability directly. Our rationale is that, assuming the ground truth is structured, models with better identifiability would still lead to more interpretable latent representation. Encouragingly, examination of the latent space extracted from these datasets indeed suggest that pi-VAE could extract interpretable and meaningful latent structure.

Monkey reaching data

We first applied our method to a previously published monkey reaching datasets ([191], kindly shared by the authors). In these experiments, Monkey C was performing a reaching task with 8 different directions, while neural activities in areas M1 and PMd were simultaneously recorded (for details, see [191]). We analyzed two sessions, and obtained similar results. We will focus on Session 1 here, and detailed results from Session 2 can be found in Appendix Figure 5.6.

For each direction, there are $\sim$ 25 trials/repeats (see Figure 5.3a). We analyzed 192 neurons from PMd area, and focused on the reaching period from go cue (defined as $t = 0$) to the end, which typically lasts for $\sim$ 1 second. We binned the ensemble spike activities into 50ms bins. We used the spike activities as observation $\mathbf{x}$, and the reaching direction as the discrete labels $\mathbf{u}$. We randomly split the dataset into 24 batches, where each batch contains at least one trial for each direction. We randomly split them into training, validation and test data $(20, 2, 2$ batches). We fit 4-dimensional latent models to the data based on pi-VAE and VAE.

Goodness of fit We first assessed the goodness of fit by examining the root-mean-square error (RMSE) of the PSTH based on the prediction of each model (see Figure 5.3a for an example neuron). Figure 5.3b,c show that pi-VAE leads to the smallest RMSE of firing rate in most neurons, followed by VAE, then tuning curve model. Next, we computed the log marginal likelihood $p(\mathbf{x})$ on the held-out test data by randomly sampling both $p(\mathbf{z}|\mathbf{u})$ and $p(\mathbf{u})$. We found that pi-VAE leads to larger mean marginal log-likelihood than VAE and tuning curve model ($-123, -123.4, -127.6$ respectively, t-test $p < 10^{-6}$). These results suggest that pi-VAE provides the best fit to the data among the alternatives.

Decoding reaching direction We wondered whether pi-VAE also provides a better encoding model of the reaching direction. We examined how well pi-VAE could decode reaching

direction, and compare the performance to a traditional method based on direction tuning curves. On held-out test data, pi-VAE achieved an average single-frame (50ms) decoding accuracy of 61%, better than 47% from tuning curve model. Examination of the time course of the decoding performance (Figure 5.3d) reveals that pi-VAE achieves 60% during the first few frames before initiation of reaching, while tuning curve model is much worse during this period. However, when reaching speed reaches its maximum (around 0.5s, Figure 5.3e), both models achieve almost perfect performance (Figure 5.3d). These observations tentatively suggest that information about the reaching direction may be encoded in different format during different phases of reaching in this task.

Structure of the latent We found that the latent variables estimated by pi-VAE exhibit clear structures. To start, the 8 reaching directions are well separated in the subspace defined by the first two latent dimensions (Figure 5.3f,h). Strikingly, the geometrical structure of the inferred latent resembles the geometry of the reaching directions. In contrast, the third and fourth latent dimensions captures the evolution of the trajectories over time, and they are only weakly informative about reaching directions (Figure 5.3g,i). Thus, the axes of the extracted latent space are easily interpretable. They provide information about how reaching direction is represented, and how neural dynamics evolve during reaching behavior. Notably, these axes were extracted automatically from pi-VAE, and no additional factor analysis techniques were applied to identify salient latent axes. Strikingly, the latent variables extracted from Session 2 show very similar structure (see Appendix Figure 5.6).

In comparison, VAE extracts a much more entangled latent representation (Figure 5.3j-m). It appears that information about reaching direction displays in a twisted fashion, and mixes with the temporal evolution of the trajectories. Note that these differences between the latent structure obtained from two methods is not simply due to the label prior. The inferred latent from pi-VAE without the label prior (*i.e.*, posterior mean of $q(\mathbf{z}|\mathbf{x})$) shows similar though a bit more diffuse latent structure, which is expected due to the observation

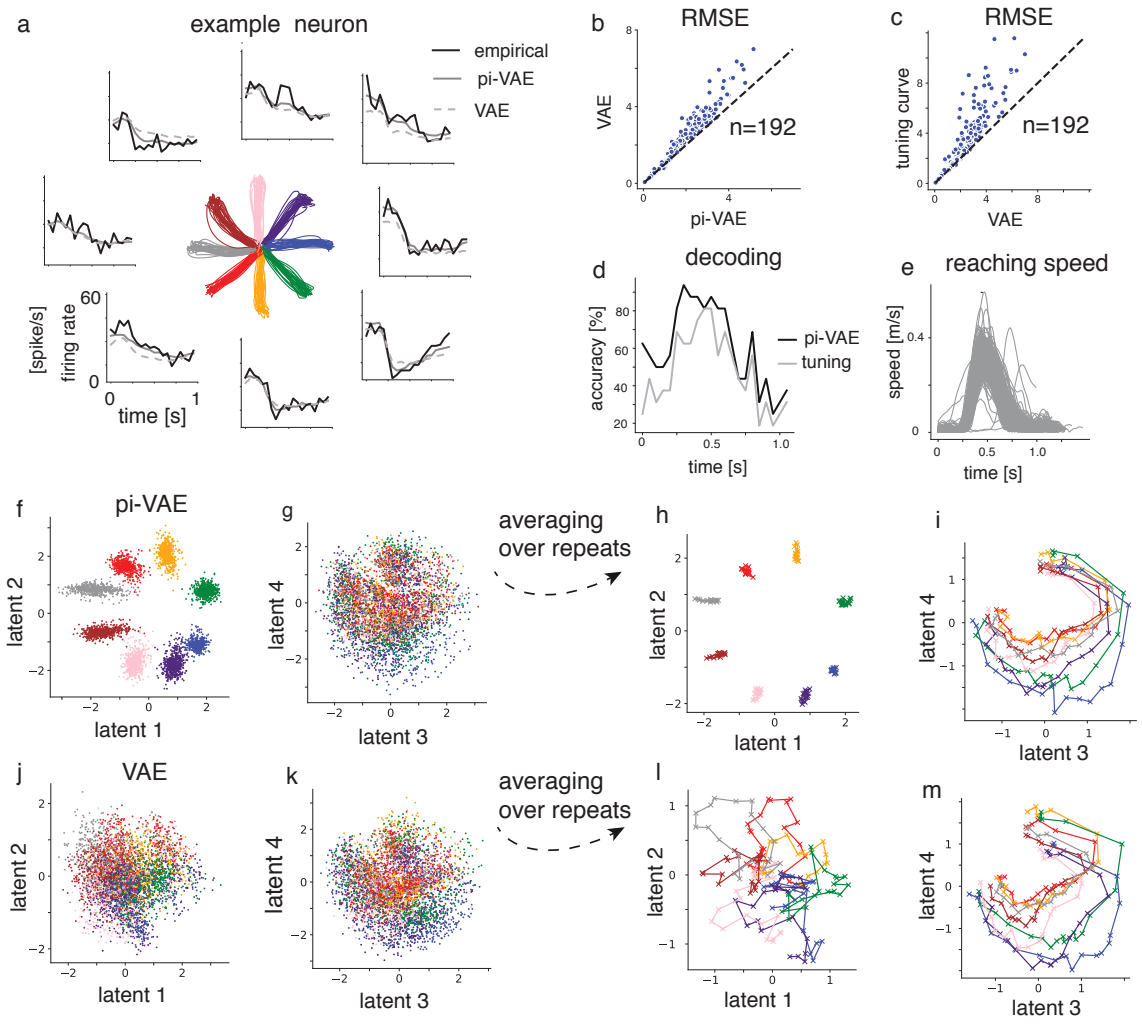

Figure 5.3: Monkey reaching data. (a) Reaching trajectories for 8 directions labeled by colors. The empirical firing rate (PSTH, black solid line), fitted rate by pi-VAE (gray solid line) and VAE (gray dashed line) for an example neuron. (b,c) Scatter plots of RMSE of fitted rate ($n = 192$ neurons) for comparing pi-VAE and VAE, as well as VAE and tuning curve. (d) Decoding accuracy as function of time on test data by pi-VAE and tuning curve model. (e) The reaching speed of the macaque for each trial. (f,g) Inferred latent based on pi-VAE, i.e.,mean of $q(\mathbf{z}|\mathbf{x}, \mathbf{u})$. (h,i) Inferred latent from pi-VAE averaged over repeats from the same reaching direction. (j,k,i,m) Similar to (f,g,h,i) for VAE. Notice the striking difference between (f) and (j).

noise (see Appendix Figure 5.7).

The nature of the neural code during primate reaching behavior is currently under heavy debate [156, 196, 160, 197, 169, 198, 199, 191]. While earlier proposals emphasized the encoding of task relevant variables [156, 196, 160, 200], some of the more recent studies instead

highlighted the importance of neural dynamics [169, 198, 201, 202]. As shown above, pi-VAE discovers latent space that exhibits striking spatial (*i.e.*, reaching direction) *and* temporal (*i.e.*, neural dynamics) structure that are separately encoded in different sub-spaces. These preliminary results may provide a way to reconcile the two prominent hypotheses [156, 198], as evidence for both hypotheses are now revealed in the same model based on the same datasets. It would be important to apply our methods to larger datasets from multiple monkeys to examine the consistency of these effects in future.

Rat hippocampal CA1 data

We next applied pi-VAE to analyze a public rat's hippocampus dataset [189, 190][1]. In this experiment, a rat ran on a 1.6m linear track with rewards at both ends (L&R) (Figure 5.4a), while neural activity in the hippocampal CA1 area was recorded ($n = 120$, putative pyramidal neurons). We focused on the data when the rat was running on the track (Figure 5.4a) and binned the ensemble spike activities into 25ms bins. We defined the rat running from one end of the track to the other end as one lap, resulting in 84 laps. We randomly split them into training, validation and test data $(68, 8, 8$ laps). We defined rat's position and running directions as continuous labels $\mathbf{u}$. We fit 2-dimensional latent models to the data for both pi-VAE and VAE.

Goodness of fit and decoding performance We found that pi-VAE again outperformed alternatives in having the lowest mean log marginal likelihood -17.7 (VAE,-17.9; tuning curve,-18.2; paired t-test, $p < 10^{-6}$). Furthermore, we decoded the animal's location on the tracking based on pi-VAE model and tuning curve model. On test data, pi-VAE achieves median absolute decoding error (MAE) of 12cm (time window = 25ms), while the tuning curve (traditional "place field" [203]) model achieves a MAE of 15cm. This indicates that for the simple purpose of constructing an effective encoding model of the animal's position on the track, pi-VAE outperforms the traditional place field model [157].

[1] `http://crcns.org/data-sets/hc/hc-11`

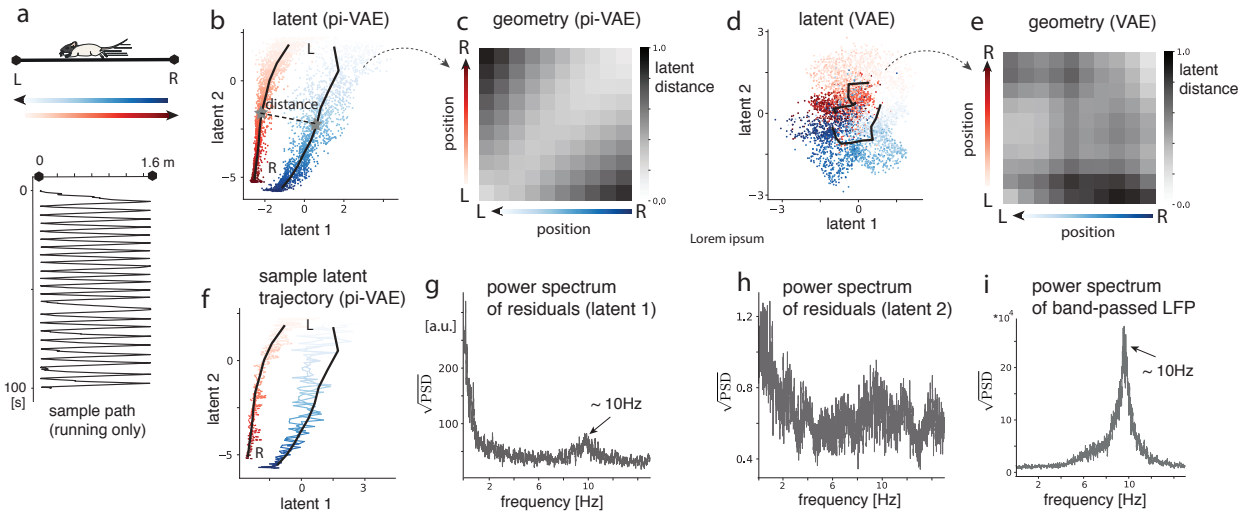

Figure 5.4: Results for hippocampus CA1 data. (a) Linear track and sample running path. The two ends are labeled as L&R. Two directions are color-coded by red and blue, and positions are coded by color saturation. (b) Inferred latent from pi-VAE. Black lines represent the mean of the latent states corresponding to position on the track for two directions. The distance between pairs of points from the two black lines is computed to quantify the latent geometry. An example pair of points are indicated using grey stars. The normalized distance for all possible pairs of points is shown in panel (c). (d,e) are defined similar to (b,c) for the VAE. Notice the striking difference between (b) and (d). (f) Two sample latent trajectories of the pi-VAE. (g,h) The power spectrum (PSD) of the residuals of the latent, given by the mean of $q(\mathbf{z}|\mathbf{x}, \mathbf{u})$ minus the mean of $p(\mathbf{z}|\mathbf{u})$. (i) PSD of the band-passed LFP in the range of 5-11 Hz.

Structure of the latent Figure 5.4b shows the latent space estimated by the pi-VAE, which exhibits overtly interpretable geometry: the collection of inferred latent states for R-to-L (blue) or L-to-R (red) running direction each forms band-like sub-manifold, and both are roughly in parallel with the second latent dimension (Figure 5.4b). The split into two sub-manifolds is consistent with the observation that place fields of CA1 neurons often have firing fields that are uncorrelated between the two travel directions ("directional" firing) [204, 114, 205]. To further quantify the geometrical relation between two sub-manifolds, we calculated the distance for pairs of points from the two branches(Figure 5.4b). This quantification for every possible pair (after binning the position into 16cm bins) is plotted in Figure 5.4c. We found that the manifold geometry across the two directions respects the geometry of the

track, in the sense that smaller physical distance on the track leads to smaller latent distance. This is likely due to that a subset of place cells have non-directional (purely spatial) place fields [204, 114, 205, 206]. Importantly, our method gives a quantitative population level characterization of the consequence of having both directional and non-directional place cells. pi-VAE without label prior shows similar but more diffuse latent structure (see Appendix Figure 5.8). In contrast, VAE results in a tangled latent representation, with the geometry not reflecting physical distance on the track (Figure 5.4d,e; also notice that striking difference between Figure 5.4b and Figure 5.4d).

To investigate whether the latent model could yield additional scientific insight, we next examined the temporal structure of the latent. Figure 5.4f plots sample trajectories, from which we observed that temporal fluctuation mainly goes along the first latent dimension, and the suggestion of rhythmic structure. We subtracted the mean of prior $p(\mathbf{z}|\mathbf{u})$ from the mean of posterior $q(\mathbf{z}|\mathbf{x}, \mathbf{u})$ to obtain the residual fluctuations. Examination of the power spectrum density (PSD) along each dimension of the residuals led to two observations: i) the temporal fluctuation is indeed concentrated on the first latent dimension, as indicated by the magnitude of the PSD; ii) the first, but not the second, dimension exhibits a striking peak at $\sim$ 10Hz. We reasoned that the second observation might be related to θ-oscillation in the local circuit, which is known to modulate the firing of CA1 neurons [207, 208, 209, 206]. We thus examined the simultaneously recorded local field potential (LFP) data during running. Indeed, we found that the θ peaked at $\sim$ 10Hz for this rat. Interestingly, the 10Hz θ-oscillation is faster than the typically reported 8 Hz average frequency [210, 211], yet is consistent with the latent structure extracted from pi-VAE.

Overall, pi-VAE extracts latent space that is clearly interpretable, with one dimension encoding position information, and the other dimension capturing temporal organization which is likely related to θ-rhythm. The observation that the position encoding and the rhythmic-like fluctuation are roughly orthogonal is particularly interesting, and is consistent with previous results from the single cell analysis suggesting that theta phase and place fields

may encode independent information [212]. Additional investigations will be needed to test these hypotheses in greater depth.

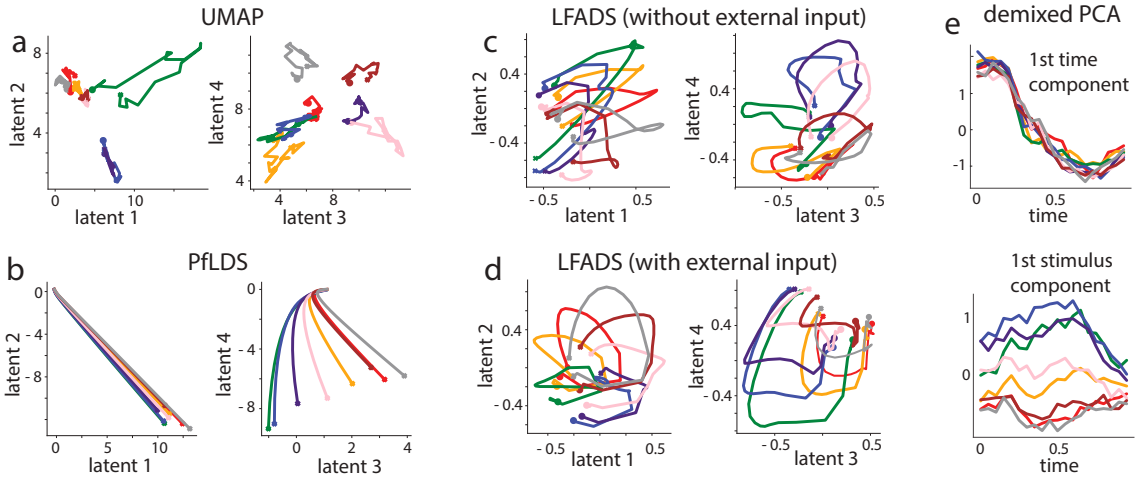

Figure 5.5: Results from alternative methods based on monkey reaching data. a) UMAP [62]. b) PfLDS [71]. c,d) LFADS [72], without and with reaching direction as an external input. e) demixed PCA [213] with the first time and stimulus component plotted. Color-coded, averaged latent trajectories corresponding to each reaching direction was plotted for each method. The filled dot and cross represent starting and ending of the trial.

5.3.3 Comparison to alternative methods

We further tested several alternative methods on the monkey reaching data, including both linear methods (demixed PCA [213]) and nonlinear methods (UMAP [62], PfLDS [71], and LFADS [72]) (see Figure 5.5). Overall we found that, while the extracted latent structures from these methods exhibited interesting characteristics, none of them resulted in fully disentangled latents. Furthermore, none of them appeared to recover the geometry of the physical reaching targets. (More analysis of the hippocampus data can be found in Appendix 5.5.5)

To start, supervised UMAP recovered latents corresponding to different directions as different clusters, but without clear representations of temporal dynamics(see Figure 5.5a). Furthermore, LFADS [72] and PfLDS [71] both led to smooth trajectories. Although the trajectories for different directions were separated in the 4-dimensional space, directions and

temporal dynamics were entangled so that it was difficult to interpret each individual latent dimension (Figure 5.5b,c,d). Demixed PCA [213] with both time and directions as labels still entangled time and directions (stimulus components change with time) to some extent (Figure 5.5e). A few methodological considerations are worth mentioning here. First, LFADS can take task variables as external inputs to the model RNN. We thus tried LFADS with or without reaching direction as external inputs (Figure 5.5c,d). Second, demixed PCA only deals with discrete task variables each with the same number of trials and each trial with the same length, and could not recover additional latent fluctuations as our method. Third, UMAP can incorporate label information for supervised learning, and we used the reaching directions as labels (Figure 5.5c,d) to make a more fair comparison. However, we found that it did not recover temporal dynamics.

5.4 Discussion

We have presented a new model framework for analyzing neural population data by integrating ingredients from latent-based and regression-based approaches. Our model pi-VAE, while being expressive and nonlinear, is constrained by additional dependence on task variables. pi-VAE generalizes recent work on identifiable VAE [49, 192, 53] to deal with spike train data. Although pi-VAE yields promising preliminary insights into the neural codes during a rat navigation task and macaque reaching task, we should emphasize that more systematic investigations based on larger datasets across different subjects will be needed to further elaborate these results.

Our method is motivated by leveraging the strength of regression-based methods and latent-based models to increase the identifiability and interpretability, a direction received little attention previously. To do so, we took advantage of the "label prior" to model the impact of task variables on neural activities along with the influence of the latent states. One potential concern is that, when incorporating too many labels, there may not be enough data to fit the model. Several previous methods exploited temporal smoothness priors to de-noise

the data, which were implemented via Gaussian process [171, 178, 179], linear [174, 175, 71] or nonlinear dynamical systems [72, 176]. Although not pursued here, adding temporal smoothness priors into pi-VAE may increase the data efficiency and further improve the performance of the model. It is also worth mentioning that although we have focused on the spike train data, our method may be modified to deal with the calcium imaging data incorporating noise models that is more appropriate to the deconvolved calcium traces [84]. Last but not least, while the current study mainly concerns the neuroscience applications of pi-VAE, some of the technical advances made here may be of interest to the machine learning community as well.

5.5 Appendix

5.5.1 Proof of identifiability for pi-VAE

Theorem 3. *Assume that we observe data sampled from pi-VAE model defined according to equation 5.1,5.2 with Poisson noise and parameters* $\boldsymbol{\theta} = (\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda})$. *Assume the following holds:*

i) The firing rate function $\mathbf{f}$ *in equation 5.1 is injective.*

ii) The sufficient statistics $T_{i,j}$ *in 5.2 are differentiable almost everywhere, and their derivatives* $T'_{i,j}$ *are nonzero almost everywhere for* $1 \leq i \leq m, 1 \leq j \leq k$.

iii) There exists $mk + 1$ *distinct points* $\mathbf{u}^0, \cdots, \mathbf{u}^{mk}$ *such that the matrix*

$$L = \Big(\boldsymbol{\lambda}(\mathbf{u}^1) - \boldsymbol{\lambda}(\mathbf{u}^0), \cdots, \boldsymbol{\lambda}(\mathbf{u}^{mk}) - \boldsymbol{\lambda}(\mathbf{u}^0)\Big)$$

of size $mk \times mk$ *is invertible, then the pi-VAE model is identifiable up to* $\sim$.

Proof. [49] has proved that the Bernoulli observation model is identifiable under the same set of assumptions. For Poisson observations with mean firing rate as λ, we can transform it to Bernoulli observations with parameter $p = 1 - \exp(-\lambda)$ by keeping the zeros and treating the positive values as ones. Because the Bernoulli model is identifiable, the Poisson model

is also identifiable. □

5.5.2 Network architecture

For both the generative models in pi-VAE and VAE, we used the following strategy to parameterize $\mathbf{f}(\cdot)$ which maps the m-dimensional latent $\mathbf{z}$ to the mean firing rate of n-dimensional Poisson observations. We first mapped the $\mathbf{z}_{1:m}$ to the concatenation of $\mathbf{z}_{1:m}$ and $t(\mathbf{z}_{1:m})$, where $t(\cdot) : \mathbb{R}^m \to \mathbb{R}^{n-m}$ is parameterized by a feed-forward neural network with a linear output and 2 hidden layers, each containing $\lfloor n/4 \rfloor$ nodes with ReLU activation function. Then we applied two GIN blocks. Same as [53], we defined the affine coupling function as the concatenation of the scale function s and the translation function t, computed together for efficiency, applied two affine coupling functions per GIN block, and randomly permuted the input before passing it through each GIN block. We defined both s, t in GIN block as mapping: $\mathbb{R}^{\lfloor n/2 \rfloor} \to \mathbb{R}^{n-\lfloor n/2 \rfloor}$. The scale function s is passed through a clamping function $0.1 \tanh(s)$, which limits the output to the range $(-0.1, 0.1)$. For affine coupling function, we have a linear output layer with and 2 hidden layers, each containing $\lfloor n/4 \rfloor$ nodes with ReLU activation function.

We modeled the prior $p_{\boldsymbol{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u})$ in pi-VAE as independent Gaussian distribution. The natural parameters $\lambda_{i,j}$ are the Gaussian means and variances. For discrete $\mathbf{u}$, we used different values of the mean and variance for different labels. For continuous $\mathbf{u}$, we parameterized the mean and spectrum decomposition of variance together by a feed-forward neural network with a linear output layer and 2 hidden layers, each containing 20 nodes with tanh activation function (the mean and variance share the 2 hidden layers). For the cases of mixed discrete and continuous labels $\mathbf{u}$, we encoded the discrete labels with a one-hot vector, and mapped it together with the continuous components to the mean and spectrum decomposition of variance using feed-forward neural network as described in the continuous case.

For the recognition model in pi-VAE and VAE, we used $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ as independent Gaussian distribution, and parameterized the mean and the spectrum decomposition of the variance

separately using feed-forward neural network with a linear output layer and 2 hidden layers, each containing 60 nodes with tanh activation function.

Code implementing the algorithms is available at https://github.com/zhd96/pi-vae.

5.5.3 Synthetic data simulations

To generate firing rate of the Poisson process from simulated latent $\mathbf{z}$, we first padded $\mathbf{z}$ with n-m zeros, then applied 4 RealNVP blocks, each containing 2 affine coupling functions with the same structure as defined in section 5.5.2 except that s does not need to have sum equal to 0 here, and we used $\lfloor n/2 \rfloor$ nodes for each hidden layer.

For discrete label simulation shown in Figure 5.2a-d, we simulated 10^4 observations, and split them into training, validation, test data ($80\%, 10\%, 10\%$ respectively). We set the batch size to be 200 during training, and trained for 600 epochs. For the continuous label simulation shown in Figure 5.2e-h, we simulated 1.5×10^4 observations. The training-validation-test split is the same as discrete label simulation. We set batch size as 300, and trained for 1000 epochs.

5.5.4 Monkey reaching data: session 2

For each reaching direction, there are ~ 35 trials (see Figure 5.6a). We analyzed 211 neurons from PMd area, and focused on the reaching period from go cue (defined as $t = 0$) to the end, which typically last for ~ 1 second. We binned the ensemble spike activities into 50ms bins. We randomly split the dataset into 34 batches, where each batch contains at least one trial from each direction. We randomly took 28 batches as training data, 3 batches as validation data and 3 batches as test data. Similar to Session 1, We fit 4-dimensional latent models to the data based on pi-VAE and VAE respectively. Results are shown in Figure 5.6.

5.5.5 Alternative methods

Monkey reaching data

For supervised UMAP[2], we set the reaching directions as labels, and embedded the high dimensional spike count data into a 4-dimensional latent space. Other parameters were set to be the default values. For PfLDS, we implemented the algorithm on our own using the same neural network architecture as in [71] (the original code provided by the authors of that chapter depends on Python Theano library, which has not been maintained for a while). We assumed a 4-dimensional latent space and Poisson observation model. We set the learning rate as 2.5×10^{-4} and trained for 1500 epochs. Each batch consisted of a single trial. The training, validation and test sets had $184, 16, 17$ trials respectively. For LFADS[3], we assumed a 4-dimensional latent space along with a Poisson observation model. We applied two versions of the model to the data, with the reaching direction as an additional input and without this input. Other parameters were set to be the default values. We pre-processed the data by discarding all the trials less than 1 second and trimming longer trials to make them 1 second long. Each batch consisted of a single trial. The training, validation and test sets had $177, 16, 16$ trials respectively. For demixed PCA[4], we pre-processed the data to make each reaching direction had the same number of trials and each trial had the same length (*i.e.*,1 second). We took time and stimulus as labels. We used 2 sets of components, each containing time, stimulus, as well as time and stimulus mixing components. Other parameters (eg. regularizer) were set to be the default values.

Hippocampus data

For supervised UMAP, we used 2-dimensional latent variables model with rat's locations as labels. Other parameters were set as default. For PCA, we used two principal components.

[2]https://umap-learn.readthedocs.io/en/latest/
[3]https://github.com/lfads/models
[4]https://github.com/machenslab/dPCA

For "PCA after LDA", we first applied LDA with rat's running directions as response and neural activities as predictors, and identified the 1-dimensional linear boundary which could separate the neural activities of two directions most. Then we projected the neural activities on this boundary using linear regression, then applied PCA with 2 principal components on the residuals. We found that the resulting latents were all less interpretable than pi-VAE (Figure 5.9), with no dimension directly representing the rat's location. Also the rhythmic-like fluctuations spanned across dimensions, rather than concentrated in one dimension (not shown).

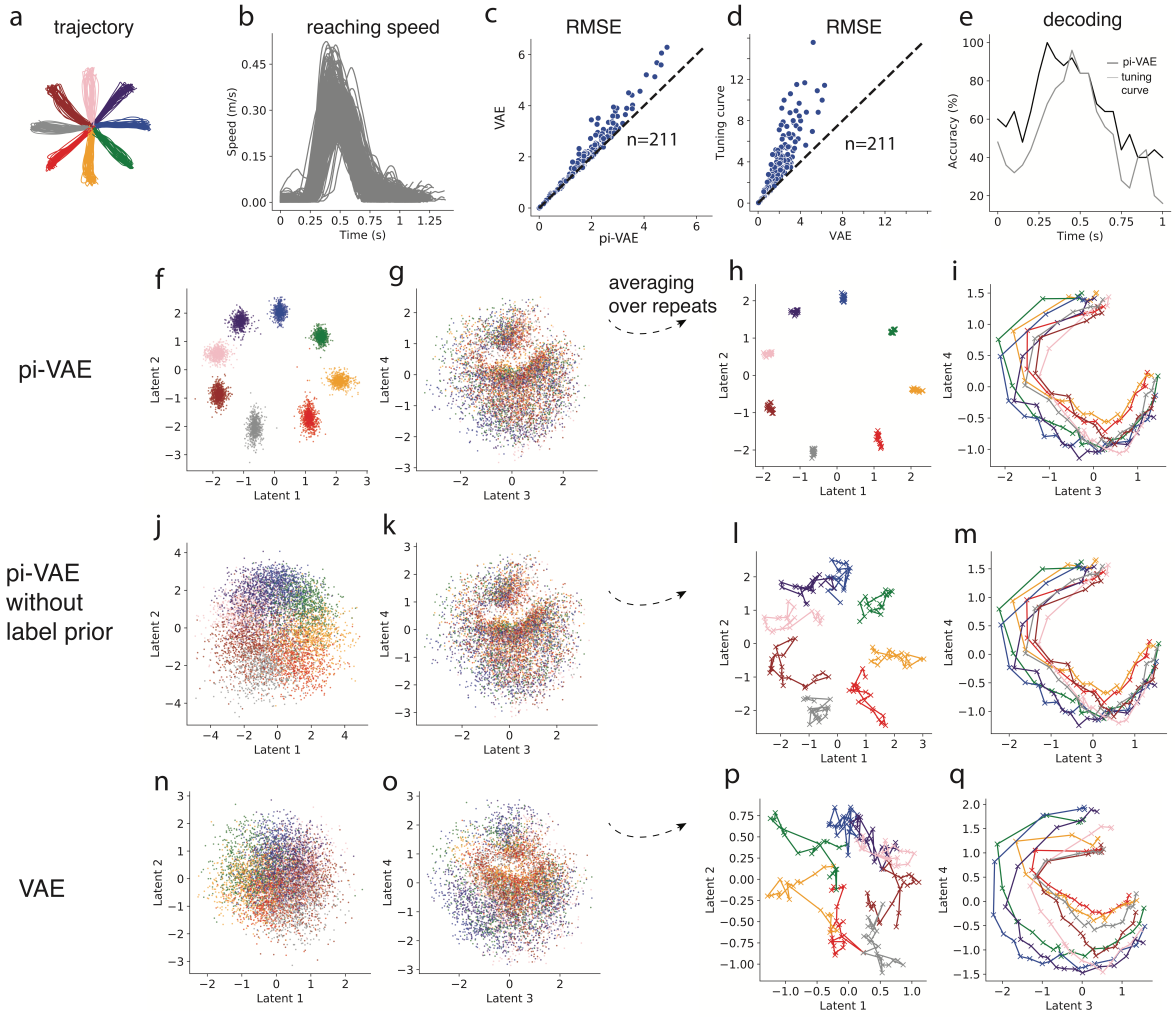

Figure 5.6: Results on monkey reaching data (Session 2). These results are similar to those obtained from Session 1 as reported in the main text. (a) The macaque's reaching trajectories for 8 directions labeled by different colors. (b) The reaching speed of the macaque for each trial. (c,d) Scatter plots of RMSE of fitted rate ($n = 211$ neurons) for comparing pi-VAE and VAE, as well as VAE and tuning curve. (e) Decoding accuracy as function of time on test data by pi-VAE and tuning curve model. (f,g) Inferred latent based on pi-VAE, i.e.,mean of $q(\mathbf{z}|\mathbf{x}, \mathbf{u})$. (h,i) Inferred latent from pi-VAE averaged over repeats from the same reaching direction. (j,k) Mean of $q(\mathbf{z}|\mathbf{x})$ from pi-VAE. (l,m) Mean of $q(\mathbf{z}|\mathbf{x})$ by pi-VAE averaging over repeats from the same reaching direction. (n-q) Similar to (f-i) for VAE.

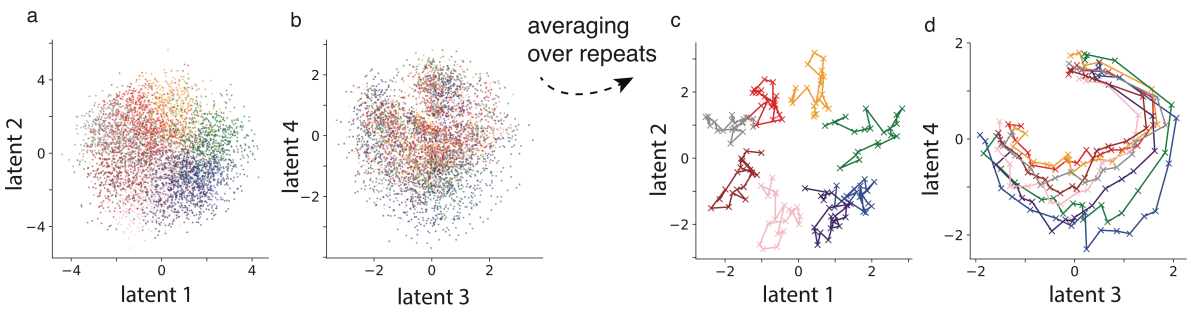

Figure 5.7: Related to Figure 5.3, on reaching data. Inferred latent without label prior using pi-VAE still are still highly structured and interpretable. The first two dimensions carry information about the reaching direction, while the third and fourth dimension mainly captures the dynamics over the time course of a trial. (a,b) Mean of $q(\mathbf{z}|\mathbf{x})$ from pi-VAE. (c,d) Mean of $q(\mathbf{z}|\mathbf{x})$ by pi-VAE averaging over repeats from the same reaching direction.

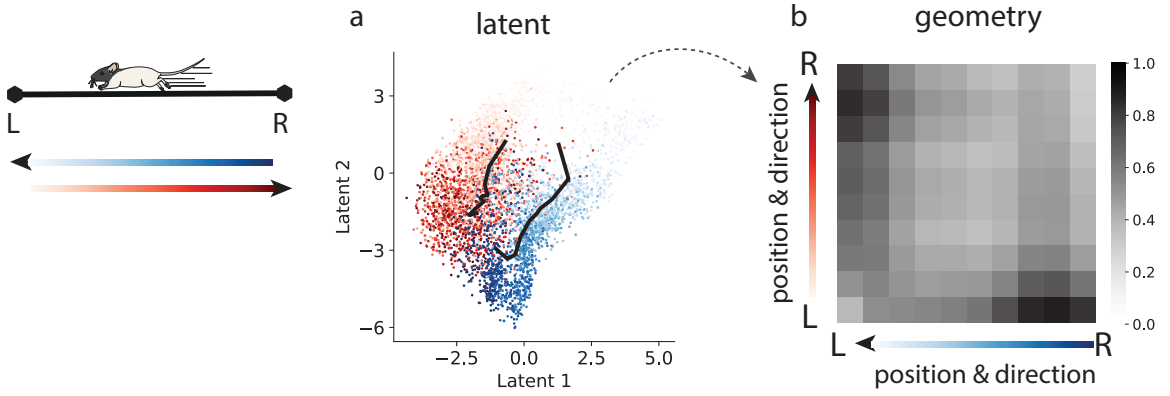

Figure 5.8: Related to Figure 5.4, on Hippocampus CA1 data. Inferred latents without the label prior using pi-VAE still exhibit clear structure, with the latent geometry respecting the geometry of the track. (a) Mean of $q(\mathbf{z}|\mathbf{x})$ from pi-VAE. Two directions are color-coded by red and blue, and positions are coded by color saturation. Black lines represent the mean of the latent states corresponding to position on the track for two directions. (b) The distance between pairs of points from the two black lines is computed to quantify the latent geometry.

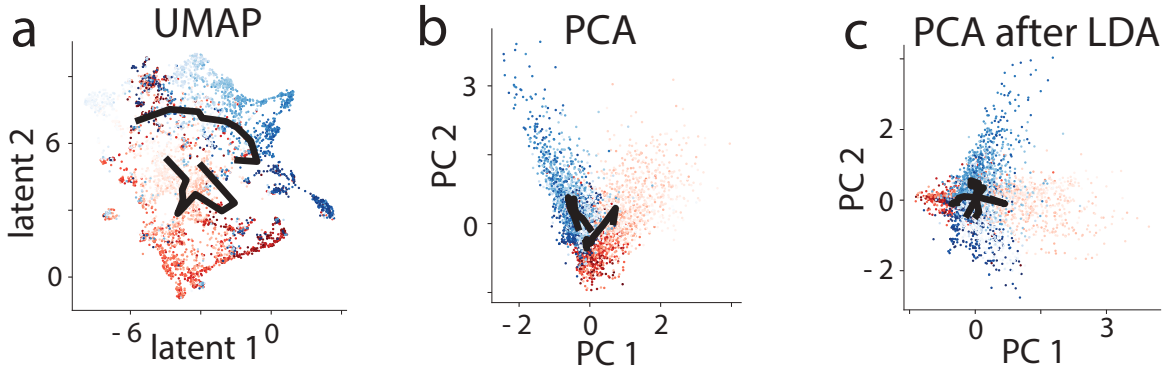

Figure 5.9: Results from several alternative methods based on hippocampus data. a) UMAP. b) PCA. c) PCA after Linear Discriminant analysis (LDA). Notice that these methods recovered more entangled representation compared to pi-VAE.

References

[1] Rafael Yuste and Winfried Denk. "Dendritic spines as basic functional units of neuronal integration". In: *Nature* 375.6533 (1995), pp. 682–684.

[2] Karel Svoboda et al. "In vivo dendritic calcium dynamics in neocortical pyramidal neurons". In: *Nature* 385.6612 (1997), pp. 161–165.

[3] Fritjof Helmchen et al. "In vivo dendritic calcium dynamics in deep-layer cortical pyramidal neurons". In: *Nature neuroscience* 2.11 (1999), pp. 989–996.

[4] Daniel A Dombeck et al. "Functional imaging of hippocampal place cells at cellular resolution during virtual navigation". In: *Nature neuroscience* 13.11 (2010), pp. 1433–1440.

[5] L. Paninski and J. Cunningham. "Neural data science: accelerating the experiment-analysis-theory cycle in large-scale neuroscience". In: *Current opinion in neurobiology* 50 (2018), pp. 232–241.

[6] JinHyung Lee et al. "YASS: Yet Another Spike Sorter applied to large-scale multi-electrode array recordings in primate retina". In: *bioRxiv* (2020).

[7] Marius Pachitariu et al. "Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels". In: *BioRxiv* (2016), p. 061481.

[8] Felix Franke et al. "An online spike detection and spike classification algorithm capable of instantaneous resolution of overlapping spikes". In: *Journal of computational neuroscience* 29.1-2 (2010), pp. 127–148.

[9] Hernan Gonzalo Rey, Carlos Pedreira, and Rodrigo Quian Quiroga. "Past, present and future of spike sorting techniques". In: *Brain research bulletin* 119 (2015), pp. 106–117.

[10] Michael S Lewicki. "A review of methods for spike sorting: the detection and classification of neural action potentials". In: *Network: Computation in Neural Systems* 9.4 (1998), R53–R78.

[11] Eran A Mukamel, Axel Nimmerjahn, and Mark J Schnitzer. "Automated analysis of cellular signals from large-scale calcium imaging data". In: *Neuron* 63.6 (2009), pp. 747–760.

[12] Ferran Diego et al. “Automated identification of neuronal activity from calcium imaging by sparse dictionary learning”. In: *2013 IEEE 10th International Symposium on Biomedical Imaging*. IEEE. 2013, pp. 1058–1061.

[13] Ryuichi Maruyama et al. “Detecting cells using non-negative matrix factorization on calcium imaging data”. In: *Neural Networks* 55 (2014), pp. 11–19.

[14] Eftychios A Pnevmatikakis et al. “Simultaneous denoising, deconvolution, and demixing of calcium imaging data”. In: *Neuron* 89.2 (2016), pp. 285–299.

[15] Marius Pachitariu et al. “Suite2p: beyond 10,000 neurons with standard two-photon microscopy”. In: *Biorxiv* (2016), p. 061507.

[16] Hakan Inan, Murat A Erdogdu, and Mark Schnitzer. “Robust Estimation of Neural Signals in Calcium Imaging”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 2905–2914.

[17] Stephanie Reynolds et al. “ABLE: An Activity-Based Level Set Segmentation Algorithm for Two-Photon Calcium Imaging Data”. In: *eNeuro* (2017).

[18] Ashley Petersen, Noah Simon, and Daniela Witten. “SCALPEL: Extracting Neurons from Calcium Imaging Data”. In: *arXiv preprint arXiv:1703.06946* (2017).

[19] Pengcheng Zhou et al. “Efficient and accurate extraction of in vivo calcium signals from microendoscopic video data”. In: *eLife* 7 (2018), e28728.

[20] E Kelly Buchanan et al. “Penalized matrix decomposition for denoising, compression, and improved demixing of functional imaging data”. In: *bioRxiv* (2018), p. 334706.

[21] Gal Mishne et al. “Automated cellular structure extraction in biological images with applications to calcium imaging data”. In: *bioRxiv* (2018).

[22] Joshua T Vogelstein et al. “Spike inference from calcium imaging using sequential Monte Carlo methods”. In: *Biophysical journal* 97.2 (2009), pp. 636–655.

[23] Joshua T Vogelstein et al. “Fast nonnegative deconvolution for spike train inference from population calcium imaging”. In: *Journal of neurophysiology* 104.6 (2010), pp. 3691–3704.

[24] Thomas Deneux et al. “Accurate spike estimation from noisy calcium signals for ultrafast three-dimensional imaging of large neuronal populations in vivo”. In: *Nature communications* 7.1 (2016), pp. 1–17.

[25] Lucas Theis et al. “Benchmarking spike rate inference in population calcium imaging”. In: *Neuron* 90.3 (2016), pp. 471–482.

[26] Johannes Friedrich, Pengcheng Zhou, and Liam Paninski. "Fast online deconvolution of calcium imaging data". In: *PLoS computational biology* 13.3 (2017), e1005423.

[27] Sean Jewell et al. "Fast nonconvex deconvolution of calcium imaging data". In: *arXiv preprint arXiv:1802.07380* (2018).

[28] Artur Speiser et al. "Fast amortized inference of neural activity from calcium imaging data with variational autoencoders". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4024–4034.

[29] Laurence Aitchison et al. "Model-based Bayesian inference of neural activity and connectivity from all-optical interrogation of a neural circuit". In: *Advances in Neural Information Processing Systems*. 2017, pp. 3486–3495.

[30] Philipp Berens et al. "Community-based benchmarking improves spike rate inference from two-photon calcium imaging data". In: *PLoS computational biology* 14.5 (2018), e1006157.

[31] Marius Pachitariu, Carsen Stringer, and Kenneth D Harris. "Robustness of spike deconvolution for neuronal calcium imaging". In: *Journal of Neuroscience* 38.37 (2018), pp. 7976–7985.

[32] David S Greenberg et al. "Accurate action potential inference from a calcium sensor protein through biophysical modeling". In: *bioRxiv* (2018), p. 479055.

[33] Eero Simoncelli, Liam Paninski, and Jonathan Pillow. "Responses with stochastic stimuli". In: *The cognitive neurosciences* (2004), p. 327.

[34] Liam Paninski. "Maximum likelihood estimation of cascade point-process neural encoding models". In: *Network: Computation in Neural Systems* 15.4 (2004), pp. 243–262.

[35] Nikhil Parthasarathy et al. "Neural networks for efficient bayesian decoding of natural images from retinal neurons". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6434–6445.

[36] Joshua I Glaser et al. "Machine learning for neural decoding". In: *Eneuro* 7.4 (2020).

[37] John P Cunningham and M Yu Byron. "Dimensionality reduction for large-scale neural recordings". In: *Nature neuroscience* 17.11 (2014), pp. 1500–1509.

[38] Ofer Mazor and Gilles Laurent. "Transient dynamics versus fixed points in odor representations by locust antennal lobe projection neurons". In: *Neuron* 48.4 (2005), pp. 661–673.

[39] Mark M Churchland et al. “Stimulus onset quenches neural variability: a widespread cortical phenomenon”. In: *Nature neuroscience* 13.3 (2010), pp. 369–378.

[40] Pierre Comon. “Independent component analysis, a new concept?” In: *Signal processing* 36.3 (1994), pp. 287–314.

[41] Suresh Balakrishnama and Aravind Ganapathiraju. “Linear discriminant analysis-a brief tutorial”. In: *Institute for Signal and information Processing*. Vol. 18. 1998. 1998, pp. 1–8.

[42] Kevin L Briggman, Henry DI Abarbanel, and William B Kristan. “Optical imaging of neuronal populations during decision-making”. In: *Science* 307.5711 (2005), pp. 896–901.

[43] Daniel Durstewitz et al. “Abrupt transitions between prefrontal neural ensemble states accompany behavioral transitions during rule learning”. In: *Neuron* 66.3 (2010), pp. 438–448.

[44] Mark A Kramer. “Nonlinear principal component analysis using autoassociative neural networks”. In: *AIChE journal* 37.2 (1991), pp. 233–243.

[45] Matthew Whiteway et al. “Characterizing the nonlinear structure of shared variability in cortical neuron populations using latent variable models”. In: *Neurons, Behavior, Data analysis, and Theory* 2.2 (2019), pp. 1–22.

[46] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[47] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. “Stochastic backpropagation and approximate inference in deep generative models”. In: *International conference on machine learning*. PMLR. 2014, pp. 1278–1286.

[48] Francesco Locatello et al. “Challenging common assumptions in the unsupervised learning of disentangled representations”. In: *arXiv preprint arXiv:1811.12359* (2018).

[49] Ilyes Khemakhem et al. “Variational Autoencoders and Nonlinear ICA: A Unifying Framework”. In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Vol. 108. PMLR, 2020, pp. 2207–2217.

[50] Irina Higgins et al. “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”. In: *International Conference on Learning Representations*. 2017.

[51] Tian Qi Chen et al. “Isolating sources of disentanglement in variational autoencoders”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 2610–2620.

[52] Christopher P Burgess et al. “Understanding disentangling in *beta*-VAE”. In: *arXiv preprint arXiv:1804.03599* (2018).

[53] Peter Sorrenson, Carsten Rother, and Ullrich Köthe. “Disentanglement by Nonlinear ICA with General Incompressible-flow Networks (GIN)”. In: *International Conference on Learning Representations*. 2020.

[54] Ilyes Khemakhem et al. “ICE-BeeM: Identifiable Conditional Energy-Based Deep Models Based on Nonlinear ICA”. In: *Advances in Neural Information Processing Systems*. 2020, pp. 12768–12778.

[55] Geoffrey Roeder, Luke Metz, and Diederik P Kingma. “On linear identifiability of learned representations”. In: *arXiv preprint arXiv:2007.00810* (2020).

[56] Mukund Balasubramanian et al. “The isomap algorithm and topological stability”. In: *Science* 295.5552 (2002), pp. 7–7.

[57] Sam T Roweis and Lawrence K Saul. “Nonlinear dimensionality reduction by locally linear embedding”. In: *science* 290.5500 (2000), pp. 2323–2326.

[58] Joshua B Tenenbaum, Vin De Silva, and John C Langford. “A global geometric framework for nonlinear dimensionality reduction”. In: *science* 290.5500 (2000), pp. 2319–2323.

[59] Debajit Saha et al. “A spatiotemporal coding mechanism for background-invariant odor recognition”. In: *Nature neuroscience* 16.12 (2013), pp. 1830–1839.

[60] Luis Carrillo-Reid et al. “Encoding network states by striatal cell assemblies”. In: *Journal of neurophysiology* 99.3 (2008), pp. 1435–1450.

[61] Laurens van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE”. In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[62] Leland McInnes, John Healy, and James Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018).

[63] George Dimitriadis, Joana P Neto, and Adam R Kampff. “T-SNE visualization of large-scale neural recordings”. In: *Neural computation* 30.7 (2018), pp. 1750–1774.

[64] Lawrence R Rabiner. “A tutorial on hidden Markov models and selected applications in speech recognition”. In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286.

[65] Rudolph Emil Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.

[66] Lauren M Jones et al. "Natural stimuli evoke dynamic sequences of states in sensory cortical ensembles". In: *Proceedings of the National Academy of Sciences* 104.47 (2007), pp. 18772–18777.

[67] Kourosh Maboudi et al. "Uncovering temporal structure in hippocampal output patterns". In: *Elife* 7 (2018), e34467.

[68] Yee Whye Teh et al. "Hierarchical Dirichlet Processes". In: *Journal of the American Statistical Association* 101.476 (2006), pp. 1566–1581.

[69] Byron M Yu et al. "Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity". In: *Journal of neurophysiology* 102.1 (2009), pp. 614–635.

[70] Jakob H Macke et al. "Empirical models of spiking in neural populations". In: *Advances in neural information processing systems* 24 (2011), pp. 1350–1358.

[71] Yuanjun Gao et al. "Linear dynamical neural population models through nonlinear embeddings". In: *Advances in neural information processing systems*. 2016, pp. 163–171.

[72] Chethan Pandarinath et al. "Inferring single-trial neural population dynamics using sequential auto-encoders". In: *Nature methods* 15.10 (2018), pp. 805–815.

[73] Yuanjun Gao et al. "High-dimensional neural spike train analysis with generalized count linear dynamical systems". In: *Advances in neural information processing systems*. 2015, pp. 2044–2052.

[74] Daniel Hernandez et al. "Nonlinear Evolution via Spatially-Dependent Linear Dynamics for Electrophysiology and Calcium Data". In: *Neurons, Behavior, Data analysis, and Theory* 3.3 (2020), p. 13476.

[75] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22.

[76] Nicholas Metropolis et al. "Equation of state calculations by fast computing machines". In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092.

[77] W Keith Hastings. "Monte Carlo sampling methods using Markov chains and their applications". In: *Biometrika* 57.1 (1970), pp. 97–109.

[78] Stuart Geman and Donald Geman. "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images". In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1984), pp. 721–741.

[79] Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[80] Michael I Jordan et al. "An introduction to variational methods for graphical models". In: *Machine learning* 37.2 (1999), pp. 183–233.

[81] Martin J Wainwright and Michael Irwin Jordan. *Graphical models, exponential families, and variational inference*. Now Publishers Inc, 2008.

[82] Herbert Robbins and Sutton Monro. "A stochastic approximation method". In: *The annals of mathematical statistics* (1951), pp. 400–407.

[83] Yoav Adam et al. "Voltage imaging and optogenetics reveal behaviour-dependent changes in hippocampal dynamics". In: *Nature* 569.7756 (2019), pp. 413–417.

[84] Xue-Xin Wei et al. "A zero-inflated gamma model for post-deconvolved calcium imaging traces". In: *Neurons, Behavior, Data analysis, and Theory* 3.2 (2020), pp. 1–21.

[85] Ding Zhou, Yuanjun Gao, and Liam Paninski. "Disentangled Sticky Hierarchical Dirichlet Process Hidden Markov Model". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2020, pp. 612–627.

[86] Ding Zhou and Xue-Xin Wei. "Learning identifiable and interpretable latent models of high-dimensional neural activity using pi-VAE". In: *Advances in Neural Information Processing Systems*. 2020, pp. 7234–7247.

[87] Julien Mairal et al. "Online dictionary learning for sparse coding". In: *International conference on machine learning*. PMLR. 2009, pp. 689–696.

[88] Johannes Friedrich et al. "Multi-scale approaches for high-speed imaging and analysis of large neural populations". In: *PLoS Computational Biology* 13.8 (2017), e1005685.

[89] Andrea Giovannucci et al. "OnACID: Online Analysis of Calcium Imaging Data in Real Time". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2378–2388.

[90] David Donoho and Victoria Stodden. "When does non-negative matrix factorization give a correct decomposition into parts?" In: *Advances in neural information processing systems*. 2004, pp. 1141–1148.

[91] Ben Recht et al. “Factoring nonnegative matrices with linear programs”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 1214–1222.

[92] Sanjeev Arora et al. “Computing a nonnegative matrix factorization – provably”. In: *Proceedings of the forty-fourth annual ACM symposium on theory of computing*. ACM. 2012, pp. 145–162.

[93] Yuanzhi Li, Yingyu Liang, and Andrej Risteski. “Recovery guarantee of non-negative matrix factorization via alternating updates”. In: *Advances in neural information processing systems* (2016), pp. 4987–4995.

[94] Nicolas Gillis and Stephen A Vavasis. “Fast and robust recursive algorithms for separable nonnegative matrix factorization”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.4 (2014), pp. 698–714.

[95] Nicolas Gillis and Robert Luce. “A fast gradient method for nonnegative sparse regression with self-dictionary”. In: *IEEE Transactions on Image Processing* 27.1 (2018), pp. 24–37.

[96] Michael E Xie et al. “High-fidelity estimates of spikes and subthreshold waveforms from 1-photon voltage imaging in vivo”. In: *Cell Reports* 35.1 (2021), p. 108954.

[97] Xiao Fu et al. “Self-dictionary sparse regression for hyperspectral unmixing: Greedy pursuit and pure pixel search are related”. In: *IEEE Journal of Selected Topics in Signal Processing* 9.6 (2015), pp. 1128–1141.

[98] Nicholas James Sofroniew et al. “A large field of view two-photon mesoscope with subcellular resolution for in vivo imaging”. In: *Elife* 5 (2016), e14472.

[99] Rongwen Lu et al. “50 Hz volumetric functional imaging with continuously adjustable depth of focus”. In: *Biomedical Optics Express* 9.4 (2018), pp. 1964–1976.

[100] Rongwen Lu et al. “Video-rate volumetric functional imaging of the brain at synaptic resolution”. In: *Nature neuroscience* 20.4 (2017), pp. 620–628.

[101] Christopher A Werley et al. “All-Optical Electrophysiology for Disease Modeling and Pharmacological Characterization of Neurons”. In: *Current Protocols in Pharmacology* (2017), pp. 11–20.

[102] Tsai-Wen Chen et al. “Ultrasensitive fluorescent proteins for imaging neuronal activity”. In: *Nature* 499.7458 (2013), pp. 295–300.

[103] Yaniv Ziv et al. “Long-term dynamics of CA1 hippocampal place codes”. In: *Nature neuroscience* 16.3 (2013), p. 264.

[104] Alon Rubin et al. “Hippocampal ensemble dynamics timestamp events in long-term memory”. In: *Elife* 4 (2015), e12247.

[105] Laura N Driscoll et al. “Dynamic reorganization of neuronal activity patterns in parietal cortex”. In: *Cell* 170.5 (2017), pp. 986–999.

[106] Carsen Stringer and Marius Pachitariu. “Computational processing of neural recordings from calcium imaging data”. In: *Current opinion in neurobiology* 55 (2019), pp. 22–31.

[107] Sean Jewell, Daniela Witten, et al. “Exact spike train inference via ℓ_0 optimization”. In: *Annals of Applied Statistics* 12.4 (2018), pp. 2457–2482.

[108] Henry Lütcke et al. “Inference of neuronal network spike dynamics and topology from calcium imaging data”. In: *Frontiers in neural circuits* 7 (2013), p. 201.

[109] Hod Dana et al. “Sensitive red protein calcium indicators for imaging neural activity”. In: *Elife* 5 (2016), e12727.

[110] George J Tomko and Donald R Crapper. “Neuronal variability: non-stationary responses to identical visual stimuli”. In: *Brain research* 79.3 (1974), pp. 405–418.

[111] DJ Tolhurst, J Anthony Movshon, and ID Thompson. “The dependence of response amplitude and variance of cat visual cortical neurones on stimulus contrast”. In: *Experimental brain research* 41.3-4 (1981), pp. 414–419.

[112] Robbe LT Goris, J Anthony Movshon, and Eero P Simoncelli. “Partitioning neuronal variability”. In: *Nature neuroscience* 17.6 (2014), pp. 858–865.

[113] Pengcheng Zhou et al. “EASE: EM-Assisted Source Extraction from calcium imaging data”. In: *bioRxiv* (2020).

[114] Thomas J Davidson, Fabian Kloosterman, and Matthew A Wilson. “Hippocampal replay of extended experience”. In: *Neuron* 63.4 (2009), pp. 497–507.

[115] Elad Ganmor et al. “Direct estimation of firing rates from calcium imaging data”. In: *arXiv preprint arXiv:1601.00364* (2016).

[116] Marcus A Triplett et al. “Model-based decoupling of evoked and spontaneous neural activity in calcium imaging data”. In: *PLoS computational biology* 16.11 (2020), e1008330.

[117] Michel A Picardo et al. “Population-level representation of a temporal sequence underlying song production in the zebra finch”. In: *Neuron* 90.4 (2016), pp. 866–876.

[118] Yuriy Mishchenko, Joshua T Vogelstein, Liam Paninski, et al. "A Bayesian approach for inferring neuronal connectivity from calcium fluorescent imaging data". In: *The Annals of Applied Statistics* 5.2B (2011), pp. 1229–1261.

[119] Jeffrey D Zaremba et al. "Impaired hippocampal place cell dynamics in a mouse model of the 22q11. 2 deletion". In: *Nature neuroscience* 20.11 (2017), p. 1612.

[120] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[121] Eleanor Batty et al. "BehaveNet: nonlinear embedding and Bayesian neural decoding of behavioral videos". In: *Advances in Neural Information Processing Systems*. 2019, pp. 15680–15691.

[122] Emily B Fox et al. "A sticky HDP-HMM with application to speaker diarization". In: *The Annals of Applied Statistics* (2011), pp. 1020–1056.

[123] Bryan Pardo and William Birmingham. "Modeling form for on-line following of musical performances". In: *Proceedings of the National Conference on Artificial Intelligence*. Vol. 20. 2. 2005, p. 1018.

[124] Matthew D Hoffman, Perry R Cook, and David M Blei. "Data-Driven Recomposition using the Hierarchical Dirichlet Process Hidden Markov Model." In: *ICMC*. Citeseer. 2008.

[125] Chia-ying Lee and James Glass. "A nonparametric Bayesian approach to acoustic model discovery". In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics. 2012, pp. 40–49.

[126] Emily Fox et al. "Nonparametric Bayesian learning of switching linear dynamical systems". In: *Advances in neural information processing systems*. 2009, pp. 457–464.

[127] Ardavan Saeedi et al. "The segmented ihmm: A simple, efficient hierarchical infinite hmm". In: *International Conference on Machine Learning*. PMLR. 2016, pp. 2682–2691.

[128] Katherine Heller, Yee Whye Teh, and Dilan Gorur. "Infinite Hierarchical Hidden Markov Models". In: *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*. Vol. 5. PMLR, 2009, pp. 224–231.

[129] Yong Song. "Modelling regime switching and structural breaks with an infinite hidden Markov model". In: *Journal of Applied Econometrics* 29.5 (2014), pp. 825–842.

[130] Anders Krogh et al. “Hidden Markov models in computational biology. Applications to protein modeling”. In: *Journal of molecular biology* 235.5 (1994), pp. 1501–1531.

[131] Gilles Celeux and Jean-Baptiste Durand. “Selecting hidden Markov model state number with cross-validated likelihood”. In: *Computational Statistics* 23.4 (2008), pp. 541–564.

[132] Jennifer Pohle et al. “Selecting the number of states in hidden markov models-pitfalls, practical challenges and pragmatic solutions”. In: *arXiv preprint arXiv:1701.08673* (2017).

[133] Matthew J Beal, Zoubin Ghahramani, and Carl E Rasmussen. “The infinite hidden Markov model”. In: *Advances in neural information processing systems*. 2002, pp. 577–584.

[134] Jayaram Sethuraman. “A constructive definition of Dirichlet priors”. In: *Statistica sinica* (1994), pp. 639–650.

[135] James Munkres. “Algorithms for the assignment and transportation problems”. In: *Journal of the society for industrial and applied mathematics* 5.1 (1957), pp. 32–38.

[136] E Pastalkova et al. “Simultaneous extracellular recordings from left and right hippocampal areas CA1 and right entorhinal cortex from a rat performing a left/right alternation task and other behaviors”. In: *CRCNS.org* (2015).

[137] Eva Pastalkova et al. “Internally generated cell assembly sequences in the rat hippocampus”. In: *Science* 321.5894 (2008), pp. 1322–1327.

[138] Scott W Linderman et al. “A Bayesian nonparametric approach for uncovering rat hippocampal population codes during spatial navigation”. In: *Journal of neuroscience methods* 263 (2016), pp. 36–47.

[139] AK Churchland et al. “Single-trial neural dynamics are dominated by richly varied movements: dataset”. In: (2019).

[140] Simon Musall et al. “Single-trial neural dynamics are dominated by richly varied movements”. In: *Nature neuroscience* 22.10 (2019), pp. 1677–1686.

[141] Matthew J. Johnson and Alan S. Willsky. “Bayesian Nonparametric Hidden Semi-Markov Models”. In: *Journal of Machine Learning Research* 14 (2013), pp. 673–701.

[142] Kevin P Murphy. “Hidden semi-markov models (hsmms)”. In: *unpublished notes* 2 (2002).

[143] Finale Doshi-Velez. "The infinite partially observable Markov decision process". In: *Advances in neural information processing systems*. 2009, pp. 477–485.

[144] Finale Doshi-Velez et al. "Nonparametric Bayesian policy priors for reinforcement learning". In: *Advances in Neural Information Processing Systems*. 2010, pp. 532–540.

[145] Finale Doshi-Velez et al. "Bayesian nonparametric methods for partially-observable reinforcement learning". In: *IEEE transactions on pattern analysis and machine intelligence* 37.2 (2013), pp. 394–407.

[146] Michael Hughes, Dae Il Kim, and Erik Sudderth. "Reliable and Scalable Variational Inference for the Hierarchical Dirichlet Process". In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. Vol. 38. PMLR, 2015, pp. 370–378.

[147] Aonan Zhang, San Gultekin, and John Paisley. "Stochastic Variational Inference for the HDP-HMM". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. Vol. 51. PMLR, 2016, pp. 800–808.

[148] Ari Pakman et al. "Neural Clustering Processes". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7455–7465.

[149] Jeffrey W Miller and Matthew T Harrison. "Mixture models with a prior on the number of components". In: *Journal of the American Statistical Association* 113.521 (2018), pp. 340–356.

[150] Kevin P Murphy. "Conjugate Bayesian analysis of the Gaussian distribution". In: *def* $1.2\sigma 2$ (2007), p. 16.

[151] Stephen Tu. "The dirichlet-multinomial and dirichlet-categorical models for bayesian inference". In: *Computer Science Division, UC Berkeley* (2014).

[152] Michael D Escobar and Mike West. "Bayesian density estimation and inference using mixtures". In: *Journal of the american statistical association* 90.430 (1995), pp. 577–588.

[153] Andrew Gelman et al. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

[154] Wilson Truccolo et al. "A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects". In: *Journal of neurophysiology* 93.2 (2005), pp. 1074–1089.

[155] Jonathan W Pillow et al. "Spatio-temporal correlations and visual signalling in a complete neuronal population". In: *Nature* 454.7207 (2008), pp. 995–999.

[156] Apostolos P Georgopoulos, Andrew B Schwartz, and Ronald E Kettner. "Neuronal population coding of movement direction". In: *Science* 233.4771 (1986), pp. 1416–1419.

[157] Kechen Zhang et al. "Interpreting neuronal population activity by reconstruction: unified framework with application to hippocampal place cells". In: *Journal of neurophysiology* 79.2 (1998), pp. 1017–1044.

[158] John O'Keefe and Jonathan Dostrovsky. "The hippocampus as a spatial map: preliminary evidence from unit activity in the freely-moving rat." In: *Brain research* (1971).

[159] Herman P Snippe. "Parameter extraction from population codes: A critical assessment". In: *Neural Computation* 8.3 (1996), pp. 511–529.

[160] Andrew B Schwartz, Ronald E Kettner, and Apostolos P Georgopoulos. "Primate motor cortex and free arm movements to visual targets in three-dimensional space. I. Relations between single cell discharge and direction of movement". In: *Journal of Neuroscience* 8.8 (1988), pp. 2913–2927.

[161] Il Memming Park et al. "Encoding and decoding in parietal cortex during sensorimotor decision-making". In: *Nature neuroscience* 17.10 (2014), pp. 1395–1403.

[162] Adam J Calhoun, Jonathan W Pillow, and Mala Murthy. "Unsupervised identification of the internal states that shape natural behavior". In: *Nature neuroscience* 22.12 (2019), pp. 2040–2049.

[163] Terence David Sanger. "Probability density estimation for the interpretation of neural population codes". In: *Journal of Neurophysiology* 76.4 (1996), pp. 2790–2793.

[164] Emery N Brown et al. "A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells". In: *Journal of Neuroscience* 18.18 (1998), pp. 7411–7425.

[165] Mike W Oram et al. "TheIdeal Homunculus': decoding neural population signals". In: *Trends in neurosciences* 21.6 (1998), pp. 259–265.

[166] Mark Stopfer, Vivek Jayaraman, and Gilles Laurent. "Intensity versus identity coding in an olfactory system". In: *Neuron* 39.6 (2003), pp. 991–1004.

[167] Ofer Mazor and Gilles Laurent. "Transient dynamics versus fixed points in odor representations by locust antennal lobe projection neurons". In: *Neuron* 48.4 (2005), pp. 661–673.

[168] Kevin L Briggman, Henry DI Abarbanel, and William B Kristan. "Optical imaging of neuronal populations during decision-making". In: *Science* 307.5711 (2005), pp. 896–901.

[169] Mark M Churchland et al. "Neural population dynamics during reaching". In: *Nature* 487.7405 (2012), pp. 51–56.

[170] Valerio Mante et al. "Context-dependent computation by recurrent dynamics in prefrontal cortex". In: *Nature* 503.7474 (2013), pp. 78–84.

[171] M Yu Byron et al. "Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity". In: *Advances in neural information processing systems*. 2009, pp. 1881–1888.

[172] Patrick T Sadtler et al. "Neural constraints on learning". In: *Nature* 512.7515 (2014), pp. 423–426.

[173] Lea Duncker and Maneesh Sahani. "Temporal alignment and latent Gaussian process factor inference in population spike trains". In: *Advances in Neural Information Processing Systems*. 2018, pp. 10445–10455.

[174] Jakob H Macke et al. "Empirical models of spiking in neural populations". In: *Advances in neural information processing systems*. 2011, pp. 1350–1358.

[175] Lars Buesing, Jakob H Macke, and Maneesh Sahani. "Spectral learning of linear dynamics from generalised-linear observations with application to neural population data". In: *Advances in neural information processing systems*. 2012, pp. 1682–1690.

[176] Lea Duncker et al. "Learning interpretable continuous-time models of latent stochastic dynamical systems". In: *arXiv preprint arXiv:1902.04420* (2019).

[177] Bede M Broome, Vivek Jayaraman, and Gilles Laurent. "Encoding and decoding of overlapping odor sequences". In: *Neuron* 51.4 (2006), pp. 467–482.

[178] Yuan Zhao and Il Memming Park. "Variational latent Gaussian process for recovering single-trial dynamics from population spike trains". In: *Neural computation* 29.5 (2017), pp. 1293–1316.

[179] Anqi Wu et al. "Gaussian process based nonlinear latent structure discovery in multivariate spike train data". In: *Advances in neural information processing systems*. 2017, pp. 3496–3505.

[180] Mohammad Reza Keshtkaran and Chethan Pandarinath. "Enabling hyperparameter optimization in sequential autoencoders for spiking neural data". In: *Advances in Neural Information Processing Systems*. 2019, pp. 15911–15921.

[181] Wei Wu et al. “Neural decoding of hand motion using a linear state-space model with hidden states”. In: *IEEE Transactions on neural systems and rehabilitation engineering* 17.4 (2009), pp. 370–378.

[182] Vernon Lawhern et al. “Population decoding of motor cortical activity using a generalized linear model with hidden states”. In: *Journal of neuroscience methods* 189.2 (2010), pp. 267–280.

[183] Alexander S Ecker et al. “State dependence of noise correlations in macaque primary visual cortex”. In: *Neuron* 82.1 (2014), pp. 235–248.

[184] I-Chun Lin et al. “The nature of shared cortical variability”. In: *Neuron* 87.3 (2015), pp. 644–656.

[185] Matthew R Whiteway and Daniel A Butts. “The quest for interpretable models of neural population activity”. In: *Current opinion in neurobiology* 58 (2019), pp. 86–93.

[186] Danilo Jimenez Rezende and Shakir Mohamed. “Variational inference with normalizing flows”. In: *arXiv preprint arXiv:1505.05770* (2015).

[187] Matthew D Hoffman and Matthew J Johnson. “Elbo surgery: yet another way to carve up the variational evidence lower bound”. In: *Workshop in Advances in Approximate Bayesian Inference, NIPS*. Vol. 1. 2016, p. 2.

[188] Alexander Alemi et al. “Fixing a broken ELBO”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 159–168.

[189] Andres D Grosmark and György Buzsáki. “Diversity in neural firing dynamics supports both rigid and learned hippocampal sequences”. In: *Science* 351.6280 (2016), pp. 1440–1443.

[190] AD Grosmark, JD Long, and G Buzsáki. *Recordings from hippocampal area CA1, PRE, during and POST novel spatial learning. CRCNS. org*. 2016.

[191] Juan A Gallego et al. “Long-term stability of cortical population dynamics underlying consistent behavior”. In: *Nature Neuroscience* (2020), pp. 1–11.

[192] Shen Li, Bryan Hooi, and Gim Hee Lee. “Identifying through Flows for Recovering Latent Representations”. In: *arXiv preprint arXiv:1909.12555* (2019).

[193] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. “Density estimation using real nvp”. In: *arXiv preprint arXiv:1605.08803* (2016).

[194] Laurent Dinh, David Krueger, and Yoshua Bengio. “Nice: Non-linear independent components estimation”. In: *arXiv preprint arXiv:1410.8516* (2014).

[195] Evan Archer et al. “Black box variational inference for state space models”. In: *arXiv preprint arXiv:1511.07367* (2015).

[196] Apostolos P Georgopoulos et al. “On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex”. In: *Journal of Neuroscience* 2.11 (1982), pp. 1527–1537.

[197] L. Srinivasan et al. “A state-space analysis for reconstruction of goal-directed movements using neural signals”. In: *Neural computation* 18.10 (2006), pp. 2465–2494.

[198] Krishna V Shenoy, Maneesh Sahani, and Mark M Churchland. “Cortical control of arm movements: a dynamical systems perspective”. In: *Annual review of neuroscience* 36 (2013), pp. 337–359.

[199] Mikhail A Lebedev et al. “Analysis of neuronal ensemble activity reveals the pitfalls and shortcomings of rotation dynamics”. In: *Scientific Reports* 9.1 (2019), pp. 1–14.

[200] Liam Paninski et al. “Superlinear population encoding of dynamic hand trajectory in primary motor cortex”. In: *Journal of Neuroscience* 24.39 (2004), pp. 8551–8561.

[201] K Cora Ames, Stephen I Ryu, and Krishna V Shenoy. “Neural dynamics of reaching following incorrect or absent motor preparation”. In: *Neuron* 81.2 (2014), pp. 438–451.

[202] Jonathan C Kao et al. “Single-trial dynamics of motor cortex and their applications to brain-machine interfaces”. In: *Nature communications* 6.1 (2015), pp. 1–12.

[203] John O’Keefe. “Place units in the hippocampus of the freely moving rat”. In: *Experimental neurology* 51.1 (1976), pp. 78–109.

[204] Francesco P Battaglia, Gary R Sutherland, and Bruce L McNaughton. “Local sensory cues and place cell directionality: additional evidence of prospective coding in the hippocampus”. In: *Journal of Neuroscience* 24.19 (2004), pp. 4541–4550.

[205] Zaneta Navratilova et al. “Experience-dependent firing rate remapping generates directional selectivity in hippocampal place cells”. In: *Frontiers in neural circuits* 6 (2012), p. 6.

[206] Kenneth Kay et al. “Constant sub-second cycling between representations of possible futures in the hippocampus”. In: *Cell* 180.3 (2020), pp. 552–567.

[207] William E Skaggs et al. “Theta phase precession in hippocampal neuronal populations and the compression of temporal sequences”. In: *Hippocampus* 6.2 (1996), pp. 149–172.

[208] David J Foster and Matthew A Wilson. "Hippocampal theta sequences". In: *Hippocampus* 17.11 (2007), pp. 1093–1099.

[209] Eva Pastalkova et al. "Internally generated cell assembly sequences in the rat hippocampus". In: *Science* 321.5894 (2008), pp. 1322–1327.

[210] IQ Whishaw and C Hippocampal Vanderwolf. "Hippocampal EEG and behavior: change in amplitude and frequency of RSA (theta rhythm) associated with spontaneous and learned movement patterns in rats and cats". In: *Behavioral biology* 8.4 (1973), pp. 461–484.

[211] György Buzsáki. "Theta oscillations in the hippocampus". In: *Neuron* 33.3 (2002), pp. 325–340.

[212] John Huxter, Neil Burgess, and John O'Keefe. "Independent rate and temporal coding in hippocampal pyramidal cells". In: *Nature* 425.6960 (2003), pp. 828–832.

[213] Dmitry Kobak et al. "Demixed principal component analysis of neural population data". In: *Elife* 5 (2016), e10989.