ABSTRACT

Title of dissertation: A LATENT VARIABLE MODELING FRAMEWORK FOR ANALYZING NEURAL POPULATION ACTIVITY

Matthew R. Whiteway
Doctor of Philosophy, 2018

Dissertation directed by: Professor Daniel A. Butts
Department of Biology

Neuroscience is entering the age of big data, due to technological advances in electrical and optical recording techniques. Where historically neuroscientists have only been able to record activity from single neurons at a time, recent advances allow the measurement of activity from multiple neurons simultaneously. In fact, this advancement follows a Moore's Law-style trend, where the number of simultaneously recorded neurons more than doubles every seven years, and it is now common to see simultaneous recordings from hundreds and even thousands of neurons.

The consequences of this data revolution for our understanding of brain structure and function cannot be understated. Not only is there opportunity to address old questions in new ways, but more importantly these experimental techniques will allow neuroscientists to address new questions entirely. However, addressing these questions successfully requires the development of a wide range of new data analysis tools. Many of these tools will draw on recent advances in machine learning and statistics, and in particular there has been a push to develop methods that can

accurately model the statistical structure of high-dimensional neural activity.

In this dissertation I develop a latent variable modeling framework for analyzing such high-dimensional neural data. First, I demonstrate how this framework can be used in an unsupervised fashion as an exploratory tool for large datasets. Next, I extend this framework to incorporate nonlinearities in two distinct ways, and show that the resulting models far outperform standard linear models at capturing the structure of neural activity. Finally, I use this framework to develop a new algorithm for decoding neural activity, and use this as a tool to address questions about how information is represented in populations of neurons.

A LATENT VARIABLE MODELING FRAMEWORK FOR ANALYZING NEURAL POPULATION ACTIVITY

by

Matthew R. Whiteway

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2018

Advisory Committee:
Professor Daniel Butts, Chair
Professor Wojciech Czaja
Professor Radu Balan
Professor Behtash Babadi
Professor James Reggia

Acknowledgments

This dissertation represents a collaborative effort between myself and many others, most notably my advisor Daniel Butts (DAB). I wrote the introduction and conclusion chapters, and the three main chapters (chapters 2-4) are adapted from published work or work that is in preparation for publication.

Chapter 2 presents a mathematical framework developed by myself and DAB. I thank the Svoboda Lab for making their data publicly available, the Collaborative Research in Computational Neuroscience (CRCNS) program for hosting the data, and the University of Maryland's Center for Comparative and Evolutionary Biology of Hearing training grant DC-00046 for funding support.

Chapter 3 resulted from a collaboration with Karolina Socha (KS) and Vincent Bonin (VB) of Neuro-Electronics Research Flanders in Belgium. KS and VB designed and conducted the experiments; myself and DAB developed the model. I also thank the Kohn Lab for making their data publicly available, and again thank CRCNS for hosting the data. This work was supported by the National Science Foundation (NSF) grant IIS-1350990.

Chapter 4 resulted from a collaboration with Ramon Bartolo (RB) and Bruno Averbeck (BA) of the National Institute of Mental Health. RB and BA designed and conducted the experiments; myself and DAB developed the model. Again, I thank the Kohn Lab and CRCNS for their efforts in supporting open-source data, and I acknowledge the University of Maryland supercomputing resources (`http://hpcc.umd.edu`) made available for conducting the research reported in this chapter. This work was supported by the NSF grant IIS-1350990.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CNN	Convolutional Neural Network
dPCA	Demixed Prinicipal Component Analysis
EM	Expectation Maximization
FA	Factor Analysis
GAM	Generalized Affine Model
GLM	Generalized Linear Model
GPFA	Gaussian Process Factor Analysis
ICA	Independent Component Analysis
i.i.d.	independent and identically distributed
L-BFGS	Limited-Memory Broyden-Fletcher-Goldfarb-Shanno
LFP	Local Field Potential
LGN	Lateral Geniculate Nucleus
LN	Linear-Nonlinear
LS	Least Squares
LV	Latent Variable
MAP	Maximum A Posteriori
MLE	Maximum Likelihood Estimate
MML	Maximum Marginal Likelihood
PCA	Principal Component Analysis
PFC	Prefrontal Cortex
PPCA	Probabilistic Principal Component Analysis
PSTH	Peri-Stimulus Time Histogram
RLVM	Rectified Latent Variable Model
ROI	Region Of Interest
S1	Primary Somatosensory Cortex
SRLVM	Stacked Rectified Latent Variable Model
STA	Spike-Triggered Average
V1	Primary Visual Cortex

Chapter 1: Introduction

1.1 Overview

Progress in our understanding of brain function has historically been limited by the difficulty of measuring the chemical and electrical processes that support neural activity. Over the previous few decades, remarkable technological progress has provided researchers with new tools for probing this activity across a wide range of spatial and temporal scales (figure 1.1). Data collected from these different modalities are beginning to offer insights into the complex computations that underlie neural functions as diverse as sensory processing, memory formation and retrieval, decision-making and motor control.

In this dissertation I analyze such data to address questions about how sensory processing is affected by internally-generated signals that are not under direct experimental control. The ability to address these questions relies on the aforementioned advances in data acquistion, as well as a broader paradigm shift in the design and execution of neurophysiology experiments.

In classic neurophysiology experiments in sensory systems (e.g. [2]), experimenters record activity from single neurons, in anesthetized animals, in response to simple stimuli (such as oriented bars and gratings in the visual domain). This exper-

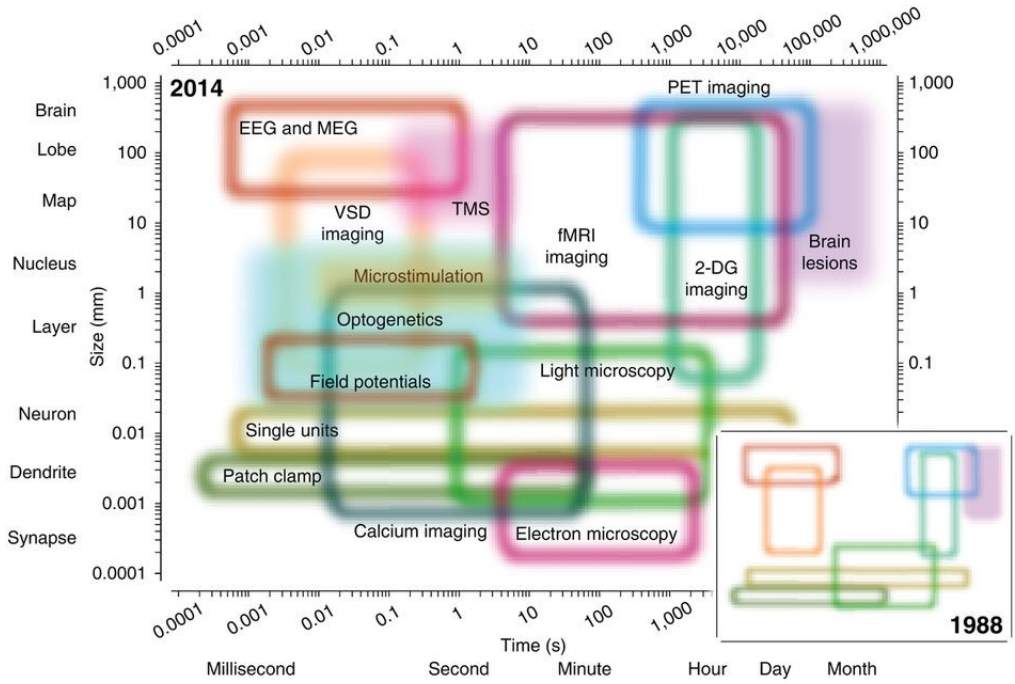

Figure 1.1: **The spatiotemporal scales of data acquisition in neuroscience.** A variety of recording modalities exist to probe neural function at a range of scales, many of which have only recently been developed. Figure from [1].

imental paradigm has persisted for many decades, and continues to enjoy widespread use. Remarkably, most of our understanding of sensory system function is derived from this highly simplified setting. However, this paradigm is ultimately limited in its ability to reveal the true underpinnings of neural function for multiple reasons. First, single neurons do not act independently (see section 1.5), and thus can only give limited insight into how populations of neurons perform relevant computations [3]. Second, anesthesia both artificially alters neural activity [4] and precludes any behavioral component to the experiment. Thus, the anesthetized preparation does not allow experimenters to make inferences about the relationship between sensory stimuli and behavior, which remains one of the fundamental open questions for systems neuroscience. Finally, artificial stimuli do not contain the complex statis-

tics of the natural stimuli that sensory systems evolved to process, which limits the scope of the conclusions drawn from these experiments (though see [5, 6]).

Modern neurophysiology experiments seek to replace one or more of these pillars of the classical paradigm to enable our understanding of neural function in more ethologically-relevant settings. Experimenters can now record neural activity from awake and behaving animals in response to natural (or at least more statistically complex) stimuli. Of particular importance to the work in this dissertation is the recently-developed ability to record activity from multiple neurons simultaneously, an ability that is growing at an exponential pace [7] (figure 1.2). Indeed, even the rate of this exponential growth is an underestimate; in chapter 4, I analyze high-quality electrophysiological data from 715 simultaneously recorded neurons, which figure 1.2 would predict possible in the year 2030.

As the field of neuroscience continues to enjoy these experimental advances, the computational tools available to analyze the resulting data remain underdeveloped. This creates an exciting opportunity for the development of new data analysis tools to uncover relevant structure in neural activity. In this dissertation I present a statistical framework for analzying such data, which finds a small number of dimensions, or latent variables, that compactly describe the high-dimensional population activity. In chapter 2, I develop the Rectified Latent Variable Model (RLVM) as a dimensionality reduction tool, and demonstrate how a linear version of this model is superior to related methods like Principal Component Analysis and Factor Analysis for understanding which aspects of the task affect neural population activity. In chapter 3, I extend the RLVM framework to develop two nonlinear latent variable

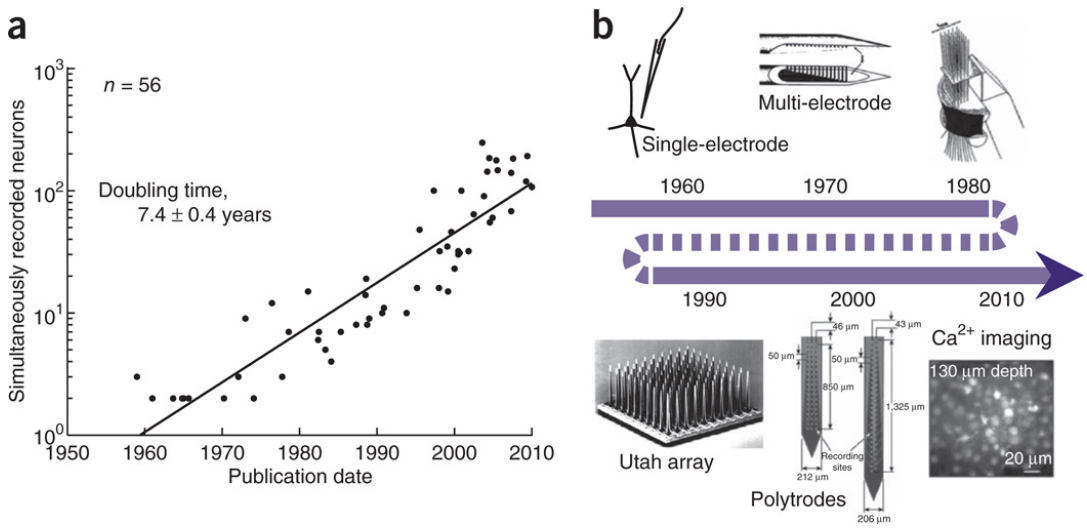

Figure 1.2: **Exponential growth in number of simultaneously recorded neurons.** **A**: The number of neurons that can be simultaneously recorded at the resolution of individual neurons (unlike the aggregate activity recorded by technology such as functional MRI) has been exponentially increasing over the span of several decades, doubling approximately every 7 years. **B**: A timeline of the recording technologies that have enabled this increase, which are discussed in section 1.3. Figure from [7].

models, the Generalized Affine Model (GAM) and Stacked RLVM (SRLVM), which are able to capture the structure of neural activity with far fewer latent variables than their linear counterparts. Finally, in chapter 4, I use this framework to develop a new decoding algorithm, and use it to address open questions about how information is represented in populations of neurons. The remainder of this introductory chapter reviews the statistical modeling work that has been developed over the previous decades to analyze both single neurons and populations of neurons, as well as the relevant concepts from neuroscience. This review will point out the limitations of these models when used to analyze neural data, in order to motivate the modeling framework developed in this dissertation.

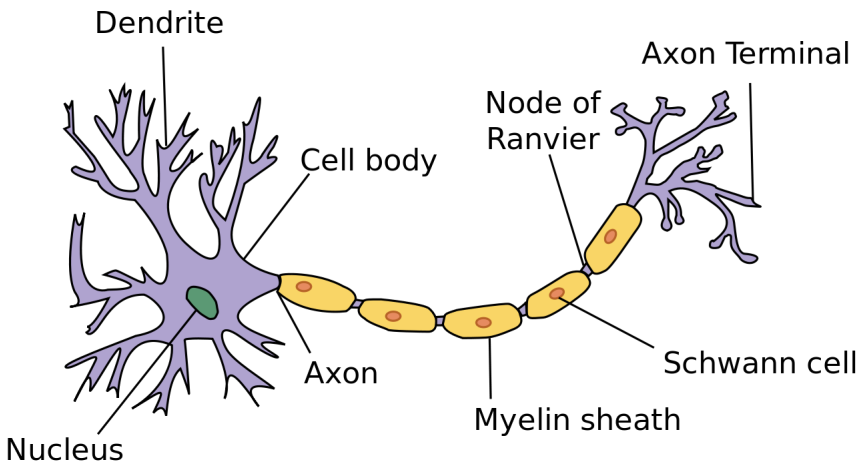

Figure 1.3: **Diagram of a neuron.** Electrical activity is generated in response to a neuron's inputs that arrive in the dendrites, and these signals are combined in the cell body (soma) and propagated down the axon to the axon terminals. This signal is then transmitted to the dendrites of other neurons through chemical or electrical connections (synapses). This image is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license, `https://commons.wikimedia.org/wiki/File:Neuron.svg`

1.2 The biological neuron: basic anatomy and physiology

A neuron is a specialized cell that is capable of rapidly transmitting information over long distances through electrical signals [8]. Though many morphological variations exist, all neurons share three main anatomical features: dendrites, the cell body (soma), and the axon (figure 1.3). Furthermore, most neurons process input signals in approximately the same manner. Biochemical mechanisms maintain a negative electrical potential across the cell membrane relative to the extracellular environment. Dendrites receive time-varying input from other neurons or through external signals from the environment, and these inputs act to either decrease (hyperpolarize) the membrane potential or increase (depolarize) the membrane potential. The soma then integrates the currents across all dendrites. If the resulting

membrane potential in the soma is sufficiently depolarized to a threshold value, the neuron generates a pulse of electrical activity called an *action potential* or *spike*. This action potential, which has an average duration of approximately 0.1 milliseconds, is propagated down the axon to the axon terminals, where it is transmitted to downstream neurons through electrical or chemical means at connections with other dendrites called synapses. Although action potential characteristics like waveform, amplitude and duration can vary among neurons, for any given neuron action potentials are highly stereotyped events [8], which is critical for the tractability of many models of neural activity.

1.3 Data acquisition methods

Though various technologies exist for recording neural activity, here I briefly introduce the two methods used to collect the data analyzed in this dissertation: extracellular recordings (chapters 3 and 4) and two-photon imaging (chapters 2 and 3). Both of these methods capture the activity of individual neurons. However, they are also invasive techniques, and have rarely been used in humans. These methods are complemented by noninvasive methods like functional magnetic resonance imaging (fMRI), electroencephalography (EEG) and magnetoencephalography (MEG), which capture signals produced by the aggregate activity of large groups ($\sim$1000s) of neurons (figure 1.1). fMRI, EEG and MEG can thus be used to record neural activity from human subjects, but at the expense of individual-neuron resolution.

1.3.1 Extracellular recordings

Extracellular recording is a popular data acquisition method that has been available for many decades (see figure 1.2B). An electrode is inserted into neural tissue (either *in vitro* or *in vivo*) near a neuron, but does not penetrate the cell membrane. Action potentials generate local changes in the electrical potential of the extracellular medium (relative to a reference electrode), which are recorded by the inserted electrode at a high sampling rate, often 10-30 samples per millisecond [9]. Intracellular recording, in which the electrode does penetrate the cell membrane, is also a useful experimental technique, but one that does not scale easily to recording from multiple neurons (and also requires a different type of electrode).

Typical extracellular recordings may pick up signals from multiple nearby neurons. In this case the resulting signals must be *spike sorted*, where spikes are assigned to individual neurons based on differing characteristics of the waveform shape, which are due to the differing orientations and distances between the recorded neurons and the electrode. Spike sorting maybe be performed in real time during the experiment or more accurately afterwards, and a host of open source and proprietary software exists for this preprocessing step.

In recent years, experimentalists have been able to steadily increase the number of neurons that can be simultaneously recorded by placing multiple electrodes on a single shank (multi-electrodes; polytrodes) or by placing multiple, single-electrode shanks on a single device (Utah array; Neuropixels [10]) (figure 1.2). This ability is crucial for understanding how the complex interactions between many neurons give

rise to neural function. In particular, the work in this disseratation relies on these large population recordings to study the role of variability in neural information processing, which is not possible with recordings from only one or a few neurons (section 1.5).

1.3.2 Two-photon imaging

Two-photon imaging is a recently developed data acquisition method that can simultaneously record the activity of thousands of individually-resolved neurons (see chapter 2).

A calcium indicator (such as GCaMP6s) is expressed by targeted neuron types in the model animal (most often mouse) using viral transduction or genetic tools. These calcium indicators are fluorescent proteins with a highly tuned absorption spectrum. This absorption spectrum changes upon binding with calcium (Ca^{2+}) ions, which enter the cell as part of the action potential generation process.

Photons from a laser can then elicit a fluorescence signal from the sample, which is typically a plane of neural tissue up to a millimeter beneath the surface of the brain (see figure 2.4 for a representative imaging plane). Importantly, the wavelength of the laser is tuned so that the energy of a single photon is half the energy required to evoke fluorescence; the calcium indicator must absorb two photons in order to emit a fluorescence photon. The probability of the calcium indicator simultaneously absorbing two photons is extremely small outside of the laser's focal point, resulting in very high spatial precision (on the order of microns). The fluo-

rescence signal is then captured by a detector, and constitutes a single pixel in the resulting image. The laser can be repeatedly scanned across the sample to produce 7-30 images per second.

Because the calcium indicators are designed to only fluoresce in the presence of Ca^{2+} ions, neurons will usually fluoresce when they generate an action potential. The duration of an action potential is on the order of 0.1 millisecond; however, the resulting fluorescence signal has a slow decay constant on the order of hundreds of milliseconds. This characteristic of two-photon imaging leads to relatively poor temporal resolution, and inferring spiking activity from these time series, a process called *spike inference*, has been the focus of much effort [11–15]. Typical post-processing of two-photon imaging experiments thus involves the extraction of fluorescence time series of individual neurons from stacks of two-dimensional images, followed by an optional spike inference step.

Despite its poor temporal resolution, two-photon imaging has become a popular tool for recording the activity of large populations of neurons. It is able to scale much better than electrophysiology, with a recent experiment imaging ∼10000 neurons simultaneously [13]. It is also becoming increasingly easy to integrate two-photon imaging with optogenetic stimulation, so that neurons can be directly stimulated based on their previous activity [16]. This ability to perform closed-loop experiments *in vivo* will open the door to many new experimental paradigms in neuroscience, as well as new analysis challenges.

1.4 Statistical models of single neurons

Mathematical descriptions of the neural activity recorded using electrophysiology or optophysiology typically take one of two forms, depending on the computational question of interest. The neural *encoding* problem is concerned with how to predict patterns of neural activity that arise through experimental manipulation of an external stimulus (a regression problem), which is addressed with the models developed in chapters 2 and 3. The neural *decoding* problem is concerned with the opposite question - how to predict a discrete external stimulus using patterns of neural activity (a classification problem), which is addressed with the decoding algorithm developed in chapter 4.

Both the neural encoding and decoding problems are typically approached using statistical models of neural activity, which attempt to discover at an abstract level how neurons represent information (often called the "neural code"). Statistical models of neural activity have recently been of great interest to the neuroscience community. Because researchers can tightly control stimulus presentation during experiments, it is possible to build complex statistical models that probe the functional relationship between sensory input and neural responses (in either direction).

This section introduces the mathematical formalism of the statistical modeling framework by considering the neural encoding problem for a single neuron. This framework will then be extended to the statistical modeling of populations of neurons in section 1.6. The framework remains essentially unchanged when considering the neural decoding problem, which is explicitly addressed in chapter 4.

To make the description of the statistical modeling framework concrete, consider the activity of a single neuron in visual cortex (model output) in response to a visual stimulus (model input). The neural activity at a particular time point t, denoted y_t, could be the number of spikes in a time window centered on t when modeling spiking data, or it could be the extracted fluorescence value from the image recorded at time t when modeling two-photon data. For an experiment with T time steps, the total activity of the neuron can be collected into the vector $\mathbf{y} \in \mathbb{R}^T$. The stimulus $\mathbf{s}_t \in \mathbb{R}^P$ at time t is defined as the luminance values for each pixel in the current stimulus, and may also include luminance values from the L previous time lags to capture the spatiotemporal patterns that drive the neural response (so that $\mathbf{s}_t \in \mathbb{R}^{LP}$)[1].

The following sections introduce various statistical models of the stimulus-response relationship between $\mathbf{s}_t$ and y_t, starting with simple linear models (section 1.4.1) and moving to increasingly complex, nonlinear models (sections 1.4.2-1.4.3). These sections also introduce the maximum-likelihood framework for learning the model parameters from data. Section 1.4.4 discusses regularization of model parameters, a crucial component of the statistical modeling framework, and then section 1.4.5 discusses approaches for evaluating the trained models.

[1]It is also possible to transform the representation of the stimulus before using it as input to the model, for example by taking the Fourier transform; see section 1.4.2 for more details.

1.4.1 Linear models

The simplest stimulus-response relationship to model is a linear one. In this case, for a single time point t the model is defined as

$$y_t = \mathbf{s}_t^\intercal \mathbf{k} + e_t \tag{1.1}$$

where $\mathbf{k}$ is the vector of parameters that maps $\mathbf{s}_t$ onto y_t and e_t is a noise term that is often assumed to be independent and identically distributed (i.i.d.) as a zero-mean Gaussian with variance σ_e^2 (denoted by $\mathcal{N}(0, \sigma_e^2)$). $\mathbf{k}$ is often referred to as the neuron's *receptive field*, and captures features of the stimulus that drive the activity of the neuron. By collecting the stimulus vectors into a matrix $S = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_T]^\intercal \in \mathbb{R}^{T \times N}$, the model can be rewritten as

$$\mathbf{y} = S\mathbf{k} + \mathbf{e} \tag{1.2}$$

where $\mathbf{e} \in \mathbb{R}^T$ is the vector of i.i.d. noise terms. Equation 1.2 is a standard linear regression problem, and the least squares (LS) solution is given by [17]

$$\hat{\mathbf{k}}_{\text{LS}} = (S^\intercal S)^{-1} S^T \mathbf{y} \tag{1.3}$$

To understand the intuition behind this equation, imagine that the stimulus is a single grayscale pixel whose values are uncorrelated in time. In this case, the $(S^\intercal S)^{-1}$ term, which captures the correlations in the stimulus, becomes a scaled version of the identity matrix cI. Furthermore, if $\mathbf{y}$ is a binary vector of spike counts, $\hat{\mathbf{k}}_{\text{LS}} = cS^\intercal \mathbf{y}$ is effectively taking the average over all stimuli that caused the neuron to spike. This model is known as the *spike-triggered average* (STA), and is a commonly used model

of neural activity across all sensory modalities because of its simple implementation and interpretation [8].

The linear model in equation 1.2 has several drawbacks when modeling spiking data. First, it allows for the possibility of negative predicted responses, but the observed spike counts in $\mathbf{y}$ are non-negative; second, the model explicitly assumes a Gaussian noise distribution, which does not match the discrete distribution of the spike counts. A more statistically principled approach to modeling this data is to choose an appropriate noise distribution and then use maximum likelihood estimation to fit model parameters [18]. [The STA can, in fact, be cast in the maximum likelihood framework, but with a noise model that is not well-matched to the data. Because of this, and because of the simple interpretation of the STA, it is never introduced in the maximum likelihood framework in the literature, and I do not do so here.]

In the maximum likelihood framework, the noise probability distribution $p(y_t|r_t)$ of the observed neural activity y_t is first defined, given a rate parameter r_t. For spiking data, $p(y_t|r_t)$ is often chosen to be the Poisson distribution [18–20], though other distributions such as Bernoulli [21], negative binomial [22] and generalized count distributions [23] have also been explored. This choice of noise distribution models the neural activity $\mathbf{y}$ as an inhomogeneous Poisson process with rate parameter vector $\mathbf{r} \in \mathbb{R}^T$. Describing the model for the neural response can then be interpreted as describing a model for the rate parameter (or equivalently, a model for the mean

response of the Poisson distribution). The linear model then becomes

$$r_t = F(\mathbf{s}_t^\intercal \mathbf{k}) \tag{1.4}$$

where $F(\cdot)$ is a function such as $F(x) = \exp(x)$ to convert $\mathbf{s}_t^\intercal \mathbf{k}$ into a non-negative value for the rate. Due to this nonlinearity, this model is referred to as the Linear-Nonlinear (LN) model [18] (and is also the definition of a generalized linear model (GLM), which are discussed in the following section).

For two-photon data, $p(y_t|r_t)$ is often chosen to be a Gaussian distribution due to empirical observations [12, 24]. Unlike the Poisson distribution, which is described by a single rate parameter, the Gaussian distribution is described by both its mean and variance. The rate r_t then models the mean of this Gaussian, and the variance is ignored (or implicitly taken into account with regularization parameters, e.g. section 2.4.1). In this case, the nonlinearity $F(\cdot)$ in equation 1.4 is unnecessary, as the mean of the Gaussian is not constrained to be non-negative, and the linear model $r_t = \mathbf{s}_t^\intercal \mathbf{k}$ is sufficient.

Once the model for r_t has been described, the parameters $\mathbf{k}$ are fit using maximum likelihood estimation. The probability of the observed responses under the model (the *likelihood*) is given by

$$p(\mathbf{y}|\mathbf{r}) = \prod_{t=1}^{T} p(y_t|r_t) \tag{1.5}$$

where the probability of the response, conditioned on the rate, is assumed to be independent across time bins. This assumption is crucial to maintaining the tractability of fitting model parameters. The likelihood is considered to be a function of the

model parameters $\mathbf{k}$, and maximizing the likelihood with respect to the model parameters defines the maximum likelihood estimate $\hat{\mathbf{k}}_{\text{MLE}}$:

$$\hat{\mathbf{k}}_{\text{MLE}} = \underset{\mathbf{k}}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{r}) \tag{1.6}$$

An equivalent formulation is to minimize the negative log-likelihood, due to the concavity of the logarithm function. One advantage of this formulation is that the product in equation 1.5 becomes a sum, which is less likely to result in numerical underflow. Furthermore, many of the common distributions used in the maximum likelihood framework come from the exponential family, so taking the logarithm is a natural and analytically simplifying choice. As a concrete example, the maximum likelihood estimate of the parameters in equation 1.4, assuming a Poisson noise distribution, is given by

$$\begin{aligned}
\hat{\mathbf{k}}_{\text{MLE}} &= \underset{\mathbf{k}}{\operatorname{argmin}} - \log p(\mathbf{y}|\mathbf{r}) && (1.7)\\
&= \underset{\mathbf{k}}{\operatorname{argmin}} - \log \prod_{t=1}^{T} p(y_t|r_t) && (1.8)\\
&= \underset{\mathbf{k}}{\operatorname{argmin}} - \sum_{t=1}^{T} \log p(y_t|r_t) && (1.9)\\
&= \underset{\mathbf{k}}{\operatorname{argmin}} - \sum_{t=1}^{T} [y_t \log r_t - r_t - y_t!] && (1.10)\\
&= \underset{\mathbf{k}}{\operatorname{argmin}} - \sum_{t=1}^{T} [y_t \log F(\mathbf{s}_t^\intercal \mathbf{k}) - F(\mathbf{s}_t^\intercal \mathbf{k})] && (1.11)
\end{aligned}$$

where moving from the third to the fourth line uses the definition of the Poisson distribution $p(y|r) = \frac{r^y}{y!}\exp(-r)$, and the $y_t!$ terms are dropped because they do not depend on the parameters $\mathbf{k}$.

The power of the maximum likelihood framework is that parameter estima-

tion is framed as an explicit optimization problem, which can be solved by standard gradient descent methods regardless of the chosen noise distribution. Another advantage is that this framework allows for arbitrarily complex models for r_t (see section 1.4.3), though there is no guarantee that local minima of the optimization problem will be equivalent to the global minimum (the two are equivalent for the LN model as long as $F(\cdot)$ is jointly convex in $\mathbf{s}$ and $\mathbf{k}$, and $\log F(\cdot)$ is jointly concave in $\mathbf{s}$ and $\mathbf{k}$, which is satisfied by both $F(x) = \exp(x)$ and $F(x) = \log(1 + \exp(x))$ [18]).

1.4.2 Generalized linear models

In the statistics literature, the generalized linear model (GLM) is a generalization of ordinary least squares that allows for non-Gaussian noise distributions [25]. The mathematical formalism for defining these models and fitting their parameters is exactly as described in the previous section. In the neuroscience literature, however, there is a historical distinction drawn between the LN model - which is, in fact, a GLM - and GLMs. LN models are usually defined as a linear transformation of the stimulus, which is then passed through the nonlinearity $F(\cdot)$ (equation 1.4). The GLMs of neuroscience use the same framework to incorporate additional predictors such as spike history terms [20], the activity from other neurons [19], local field potentials [26], and kinematic information of limb movements [27]. Furthermore, regularization is more commonly used in GLMs, not just in frequency but also in variety (section 1.4.4).

GLMs have become extremely popular tools in neuroscience due to their ease

of implementation and interpretability. Many standard statistical software packages include functions for building and fitting GLMs, lowering the barrier to their use for many experimentalists. Interpretation of the resulting model parameters is also straightforward - each parameter, after passing through the function $F(\cdot)$, denotes how much the predicted output will change given a unit increase in the predictor of interest (e.g. a single pixel of a visual stimulus).

However, most neurons are highly nonlinear, especially as they become further removed from the sensory periphery [28], and GLMs will fail to capture nonlinear features of the neural response. Therefore, understanding the complex nonlinear processing that underlies most brain function requires nonlinear modeling methods.

The first efforts in this direction applied nonlinear transforms to the input data to obtain a more linear relationship with the neural response, and linear models like GLMs could then be used. This process is called "linearizing" the encoding model [29]; examples in vision include applying the Fourier transform [30] or a wavelet transform [31] to the images and using the resulting coefficients as predictors rather than the raw pixel values; examples in audition include computing the spectrogram [32] or extracting a speech envelope [33] of the audio signal. Though linear models can recover nonlinear relationships through this approach, it also introduces the new challenge of choosing an appropriate nonlinear feature space. Though experimentalists may have some intuition about relevant nonlinear feature spaces in early stages of sensory processing, the complex features that are necessary for higher-level sensory processing are at present much more difficult to define. Increases in computational power and advances in statistical modeling techniques

now make it possible to learn the nonlinear feature space directly from the data, an approach that is discussed in the next section.

1.4.3 Hierarchical models

Nonlinear models of neural activity have a long history in neuroscience. Indeed, the need to incorporate nonlinearities was recognized by Hubel and Wiesel with their discovery of "complex cells" in the cat primary visual cortex (V1) in the early 1960s [34]. "Simple cells", which they had discovered several years previously in the same brain region [2], respond to Gabor-like stimuli with a particular orientation, spatial frequency and phase (figure 1.4B). Complex cells, on the other hand, respond to Gabor-like stimuli of a given orientation and spatial frequency, but are phase-invariant. This phase invariance was formally captured by the "energy model" of Adelson and Bergen in the 1980s [35], which described complex cell activity using two linear filters $\mathbf{k}_1$ and $\mathbf{k}_2$ with a 90 degree phase offset (a *quadrature pair*), and passing the resulting filtered stimulus through quadratic nonlinearities before adding the results to arrive at the estimate of the neural activity (figure 1.4):

$$y_t = (\mathbf{s}_t^\intercal \mathbf{k}_1)^2 + (\mathbf{s}_t^\intercal \mathbf{k}_2)^2 \tag{1.12}$$

It is important to point out that the energy model, in its original form, was a conceptual model of V1 complex cell function, and not a statistical model. However, the energy model can now be easily fit to data using the maximum likelihood framework [20]. The ability to fit the energy model to data illustrates that nonlinearities can be introduced into the model structure, while still maintaining tractable pa-

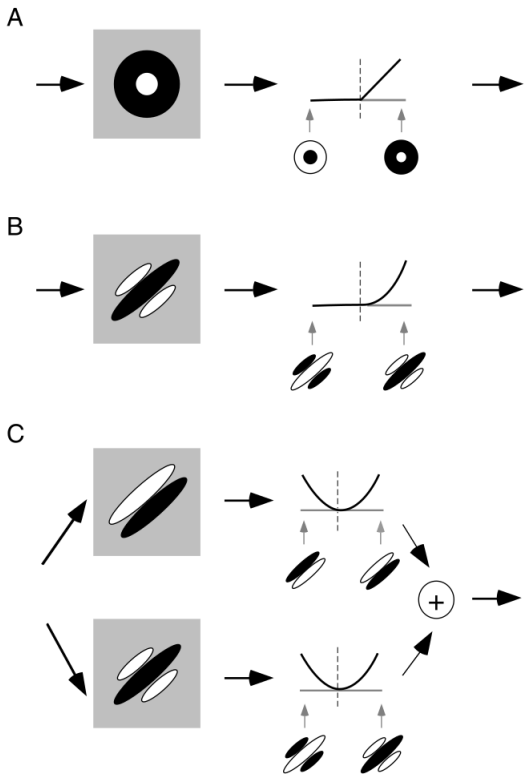

Figure 1.4: **Models of early visual processing.** Responses of visual neurons are modeled as the inner product of an arbitrary stimulus with the neuron's receptive field (gray squares), and the result is then passed through a nonlinearity to arrive at the neuron's estimated firing rate. **A**: Model of a retinal ganglion cell or LGN cell with a center-surround receptive field structure. **B**: Model of a V1 simple cell. **C**: Model of a V1 complex cell, which requires summing the result of two *filter + nonlinearity* operations to arrive at the final firing rate. Figure from [36].

rameter estimation through numerical optimization methods. As stated previously, the introduction of these nonlinearities often results in a nonconvex optimization problem with many local minima, but nevertheless these models have still found wide use in neuroscience.

Many statistical models of neural function now incorporate a single nonlinear layer of processing into the GLM, which is known as the Nonlinear Input Model in the neuroscience literature [20, 37], and also known as the Generalized Additive

Model in the statistics literature [38]:

$$r_t = F\left[\sum_{i=1}^{I} w_i f_i(\mathbf{s}_t^\intercal \mathbf{k}_i)\right] \tag{1.13}$$

The functions $f_i(\cdot)$ can be any static nonlinearity, and a wide range have been explored in the neuroscience literature, including quadratic [39, 40], rectified linear [20], hyperbolic tangent [41,42], and nonparametric [20,37,43] functions. For each of the I terms in the sum, the value of the inner product between the stimulus and the linear receptive field $\mathbf{k}_i$ is passed through the function $f_i(\cdot)$, just as the LN model of equation 1.4. The outputs of these LN models are then linearly combined and passed through the function $F(\cdot)$ as the estimate of neural activity. This two-step procedure is a hierarchical process in which linear stimulus features are extracted at the first stage, and then nonlinearly combined to produce the output. From the machine learning perspective, this model is exactly equivalent to a neural network with a single hidden layer.

The recent success of deep learning in a variety of fields has motivated computational neuroscientists to explore the potential of neural networks with more than a single hidden layer, particularly convolutional neural networks (CNNs) in the visual domain [28, 44, 45]. CNNs, despite their increased complexity and parameter count, can still be fit using the maximum likelihood framework, and in the past several years have achieved state of the art performance in modeling visual neurons in retina [46, 47], V1 [48–51] and V4 [52] (though research in this direction has a long history [53]).

The success of CNNs in modeling sensory neurons is perhaps unsurprising,

given that their architecture was originally inspired by Hubel and Wiesel's ideas about hierarchical computations in visual cortex [54]. These ideas about hierarchical computation have not only been discussed, but embraced within the sensory neuroscience community for some time [55]. However, the ability to fit these models is relatively new, and is due to a confluence of several factors: new techniques in machine learning for efficient parameter estimation (which still falls within the maximum likelihood framework, e.g. the use of stochastic gradient descent, non-saturating nonlinearities, etc.), increased computing power, and the aforementioned advances in experimental neuroscience that allow for the collection of the large amounts of data that are needed to fit these models.

Hierarchial models of neural responses can contain millions of parameters, which must be estimated from data. When fitting models in such high dimensional spaces, care must be taken to avoid overfitting the limited training data. Overfitting refers to the situation when a model fits noise in the data, rather than meaningful signal [17]. The problem of overfitting is relevant for simpler models as well - even a simple LN model can contain thousands of parameters, which will be prone to overfitting when using the amount of data that is recorded in a typical neuroscience experiment. To partially address this concern, *regularization* is a technique for biasing model parameters in order to decrease the regions of parameter space that contain potential solutions, which is the topic of the next section.

1.4.4 Regularization

One of the most popular general-use regularization techniques is L_2 regularization, known as "ridge regression" in the linear regression setting and "weight decay" in the neural networks literature. L_2 regularization adds a term to the model's cost function that penalizes the squared magnitude of the parameters:

$$\hat{\mathbf{k}} = \underset{\mathbf{k}}{\operatorname{argmin}} - \log p(\mathbf{y}|\mathbf{r}) + \lambda \|\mathbf{k}\|_2^2 \tag{1.14}$$

where $\|\mathbf{k}\|_2^2 = \sum_{p=1}^{P} k_p^2$ is the L_2 norm of the vector $\mathbf{k}$. λ is a hyperparameter that controls the contribution of the regularization term to the overall cost function. Appropriate selection of hyperparameters is an important issue that will be addressed in section 1.4.5.

In the machine learning literature, regularization terms are typically motivated by heuristic considerations. However, the statistical modeling framework naturally incorporates these regularization terms using *maximum a posteriori* (MAP) estimates rather than maximum likelihood estimates. In this framework, prior knowledge about the distributions of the parameters $p(\mathbf{k})$ is introduced using Bayes Rule. Under this rule the posterior distribution of the parameters given the data is defined as

$$p(\mathbf{k}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{k})p(\mathbf{k})}{p(\mathbf{y})} \tag{1.15}$$

Again taking the negative logarithm for numerical convenience, and ignoring the term in the denominator (which does not depend on model parameters), the MAP

estimate is given by:

$$\hat{\mathbf{k}}_{\text{MAP}} = \underset{\mathbf{k}}{\text{argmin}} - \log p(\mathbf{k}|\mathbf{y}) \tag{1.16}$$

$$= \underset{\mathbf{k}}{\text{argmin}} - \log \left[p(\mathbf{y}|\mathbf{k})p(\mathbf{k})\right] \tag{1.17}$$

$$= \underset{\mathbf{k}}{\text{argmin}} - \log p(\mathbf{y}|\mathbf{k}) - \log p(\mathbf{k}) \tag{1.18}$$

Within this framework, L_2 regularization is equivalent to the statistical assumption that the individual parameters in the vector $\mathbf{k}$ are i.i.d. $\mathcal{N}(0, \sigma_k^2)$, which can be written as

$$p(\mathbf{k}) = \prod_{p=1}^{P} \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{k_p^2}{2\sigma_k^2}\right) \tag{1.19}$$

and taking the negative logarithm of this term results in

$$-\log p(\mathbf{k}) = \frac{1}{2\sigma_k^2} \sum_{p=1}^{P} k_p^2 + \text{const} \tag{1.20}$$

which is the same as the L_2 penalty term in equation 1.14 with $\lambda = 1/(2\sigma_k^2)$, and therefore the regularization hyperparameter λ is related to the variance of the prior distribution on $\mathbf{k}$.

Another popular type of regularization term is L_1 regularization, which approximately enforces sparseness in the parameters [56]; this type of regularization is equivalent to a Laplace prior distribution in the MAP framework. A wide array of regularization terms have been explored in the neural modeling literature, including sparseness (L_1) [20, 57, 58], smoothness [20, 59, 60], and locality [61, 62] of receptive field structures.

1.4.5 Evaluating model performance

Once a model has been selected and its parameters fit, it must be evaluated to determine how well it captures the relationship between the input and output data (e.g. stimulus and neural response for neural encoding models). Several evaluation measures are commonly used, including the negative log-likelihood and the coefficient of determination (R^2) for neural encoding models, or the fraction of correctly classified stimuli for neural decoding models. This evaluation should be performed on data that was not used to train the model, in order to test how well the model generalizes to new data.

A standard way to validate model performance is by using k-fold cross validation [29]. In this procedure, the data is split into k equally-sized pieces, or folds. The model parameters are fit using $k-1$ of the folds (the training data), then evaluated on the data in the remaining fold (the validation data). The model fitting and evaluation process is repeated k times, with each fold being used once for validation. The result is k values of the evaluation measure, which can provide an average performance measure with empirical error bars.

The k-fold cross validation process is also a convenient way (albeit a computationally intensive one) to perform model selection (e.g. choosing hyperparameter values), and is called "nested k-fold cross validation" [29]. To do this, for each training data set of $k-1$ folds (one iteration of the "outer loop"), another round of cross valiation is performed (the "inner loop"). The training data is divided into a perhaps different number of m folds. Full m-fold cross validation is performed using the

training data across a range of hyperparameter values, and the hyperparameter that results in the best model (according to the appropriate evaluation measure) is then chosen for final evaluation on the validation data set. In this way, hyperparameters can be chosen using the data, but without artificially inflating performance measures by choosing the value using the validation data. Variations on this procedure are used throughout this dissertation, and the evaluation measures and validation process for individual models are discussed in the *Methods* section of each chapter.

1.5 Variability in single neuron responses

The hierarchical models discussed in section 1.4.3 are now the most sophisticated statistical models of visual and auditory processing, and ongoing work will make them invaluable tools for studying the complex nonlinear computations that underlie sensory processing. However, as currently implemented, these hierarchical models (also referred to as *feed–forward* models) implement deterministic mappings from stimulus to response, and therefore fail to capture one of the most important and ubiquitous phenomena in neuroscience: neural variability.

Neural variability is defined as the variability in neural responses across repeated presentations of the same stimulus. This variability is typically smaller in early sensory areas, and has allowed for highly successful modeling efforts in areas like retina [19, 63, 64] and LGN [65, 66]. However, this variability tends to increase along the sensory processing hierarchy [67–69], which presents yet another limiting factor in the ability to accurately model higher-order sensory neurons (beyond

their complex, hierarchical processing). Understanding this variability in neural responses is an important goal for systems neuroscience, because response variability limits the information that a neural population can contain about an external stimulus [70], which further limits the precision of the downstream processing that ultimately leads to behavior.

To properly approach the topic of neural variability, it is important to consider what variability means in this context. For many years this variability was considered to be "noise": an unfortunate byproduct of imprecise biological processes such as synaptic transmission failures [71]. In contrast, it has also been observed that cortical neuron responses are extremely precise *in vitro* [72]. Is it possible to reconcile these disparate observations? In the words of Pierre-Simon Laplace, randomness is only a measure of our "ignorance of the different causes involved in the production of events" [73]. In other words, perhaps this "noise" is actually just unexplained (though potentially explainable) variability that has functional significance to the neural computations. Barlow recognized this possibility as far back as 1972, writing that "the apparently erratic behavior [of neural responses] was caused by our ignorance, not the neuron's incompetence" [74].

Attempts to understand variability in neural responses have been greatly aided by the experimental technologies discussed in section 1.3: when measuring single-neuron activity, it is impossible to separate noise from activity that is not locked to the stimulus or other experimentally controlled factors. Simultaneous recordings from multiple neurons unlock the potential to disentangle structured variability from true noise, and address questions about the functional role of this structured

variability in neural computations. An early and still incredibly influential approach to understanding neural variability is through studying the correlated responses of pairs of neurons (section 1.5.1), which has received significant attention from both experimental [75] and theoretical [70] neuroscience. Recent large-scale recordings have also revealed that variability is shared among not just pairs of neurons, but larger groups as well (section 1.5.2). The following sections briefly review some new insights that are emerging about the structure of neural variability, and its possible functional relevance.

1.5.1 Noise correlations

Measuring correlations between the activity of pairs of neurons provides one of the simplest ways to describe structure in neural population activity. Two notions of correlation are often used in the neuroscience literature, and are typically measured by repeatedly presenting a range of stimuli while recording activity from two or more neurons (e.g. presenting oriented gratings while recording from V1; figure 1.5A). *Signal correlations* refer to the similarity in stimulus tuning of two neurons, and is calculated by averaging responses across identical trials for each neuron, and then calculating the correlation of these mean responses between the pair (figure 1.5C). *Noise correlations* refer to the correlations between neurons across repeats of a single stimulus (figure 1.2B). Because signal correlations average over repeats of identical trials, they can be studied with sequential recordings of single neurons rather than simultaneous recordings, and have thus been well-studied in the literature [76].

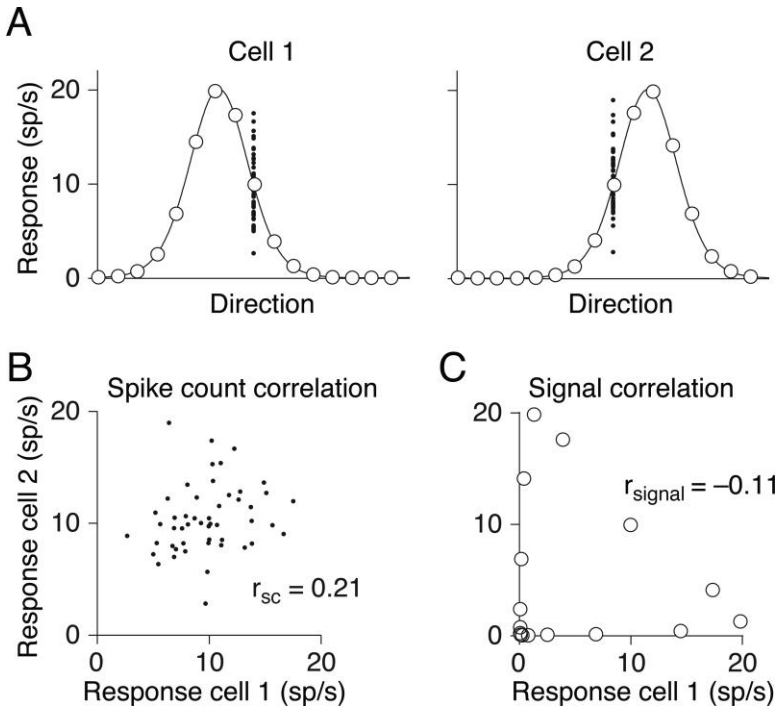

Figure 1.5: **Signal and noise correlations.** **A**: Tuning curves for two hypothetical direction-selective neurons. Open circles show mean responses to different directions of motion and small points show responses to individual presentations of a stimulus at a particular direction. **B**: Spike count or "noise" correlation (r_{SC}) measures the correlation between fluctuations in responses to the same stimulus. Here, each point represents the response of the two neurons on one presentation of an individual stimulus. **C**: Signal correlation (r_{signal}) measures the correlation between the two cells' mean responses to different stimuli. Each point represents the mean response to a given direction of motion. Because the responses of cell 2 increase to a range of motion directions in which the responses of cell 1 decline, signal correlation is negative. Figure and caption from [75].

Much recent experimental work has focused on noise correlations [75] because, unlike signal correlations, they are dependent on a wide range of ethologically relevant factors, including stimulus identity [77, 78], behavioral context [79, 80], brain state [81, 82], learning [83–86], attention [86–88] and adaptation [89]. Several studies have demonstrated that attention and learning can lead to increased behavioral performance through the reduction of noise correlations [86–88], demonstrating that there is indeed a relationship between the structure of neural variability and behav-

ior, though the nature of this relationship is still poorly understood.

Additionally, advances in both theoretical [90–95] and experimental [89,96–98] work have demonstrated that noise correlations can affect how much information neural activity contains about an external stimulus. Thus noise correlations, while being conceptually simple and experimentally easy to measure, have the potential to dramatically affect how well populations of neurons can extract that information to perform useful computations.

1.5.2 Global fluctuations

The majority of the literature concerning noise correlations analyzes small populations (1s to 10s) of neurons [75]. Experimental access to larger populations (100s to 1000s) of simultaneously recorded neurons presents new opportunities for empirically probing how neural variability is manifested beyond pairs of neurons. Indeed, data from a wide variety of recording modalities increasingly reveal the presence of variability that is shared not just among pairs of neurons, but among much larger groups of neurons as well (and hence termed *global fluctuations*). This shared variability has been measured at the individual-neuron resolution using multi-electrode arrays [81] and two-photon imaging [99], and also at a lower spatial resolution across larger brain regions using local field potentials [26], voltage-sensitive dyes [100], and wide-field imaging [85].

Many studies now suggest that this shared variability has a relatively simple, low-dimensional structure [67, 81, 82, 96, 97, 101, 102], which has important implica-

tions for both modeling variability and understanding its role in sensory processing [103–105]. This observed low-dimensional structure has motivated the use of dimensionality reduction techniques (also called *latent variable models*) for analyzing large populations of neurons. These techniques extract a small number of factors (latent variables) that describe the high-dimensional neural activity, and can also parsimoniously explain the observed pairwise correlations [80, 93, 96, 102]. These factors are easier to interpret and analyze than the full set of pairwise correlations, and have led to both theoretical and experimental work exploring their effect on sensory processing [94–96, 98].

The efficacy of these latent variable models in describing the structure of high-dimensional neural activity has made them important tools for both theorists and experimentalists, and has spurred their continued development for neuroscientific applictations. The following section introduces a range of latent variable models, and describes their strengths and weaknesses with respect to their use in neuroscience, which motivates the development of the models introduced in this dissertation.

1.6 Latent variable models of neural populations

Latent variable models are statistical models that aim to describe the complex statistical relationships in high-dimensional data with a much smaller number of factors, or *latent variables* $\mathbf{z}_t$. The encoding models of section 1.4 modeled the probability distribution $p(y_t|\mathbf{s}_t)$, where $\mathbf{s}_t$ was a known and experimentally controlled quantity. Latent variable models, on the other hand, attempt to model the prob-

ability distribution $p(\mathbf{y}_t|\mathbf{z}_t)$, where $\mathbf{z}_t$ is unknown (and thus must be inferred from the data) and $\mathbf{y}_t$ represents the activity of many neurons, rather than just one.

The computational neuroscience community has long embraced the Bayesian approach to latent variable modeling, wherein a probabilistic model defines the transformation from latent variables to neural activity, and the posterior distribution of the latent variables given the observed neural activity $p(\mathbf{z}|\mathbf{y})$ is inferred (sometimes approximately). This approach typically incorporates temporal structure in the evolution of the latent variables (called a *dynamical prior*), which has been modeled using linear dynamical systems [106–110], Gaussian processes [81,111,112], switching dynamical systems [113, 114] and recurrent neural networks [115].

An alternative to the Bayesian approach uses direct optimization of a cost function, which is designed to preserve features of interest in the data in order to learn a mapping from low-dimensional factors to high-dimensional neural activity [116]. In this approach, models are typically constrained by enforcing a static statistical structure on the distribution of the latent variables, for example defining them to be uncorrelated (PCA, tensor factorizations) [117–120], independent and Gaussian (FA) [121–123], or independent and non-Gaussian (ICA) [124].

The methods developed in this dissertation follow the direct optimization approach (though the initial formulation of the RLVM in chapter 2 is Bayesian in spirit; see section 2.4.1). This choice is due to several factors, including the difficulty of defining and fitting appropriate generative models in the Bayesian framework and the greater flexibility enjoyed by the direct optimization approaches. The dynamical priors of most Bayesian models reflect their development for the analysis of motor

cortical activity and brain-machine interfaces [23, 109, 111, 115]. Much of the data analyzed in this dissertation comes from primary sensory areas, where the use of dynamical priors is not well motivated, and it is also currently unclear what would constitute a more appropriate class of priors. Furthermore, the Bayesian framework is typically constrained by the tractability of the inference problem. Nevertheless, recent advances in variational inference adopt ideas from deep learning to make inference more tractable [125–128], and the combination of the two approaches will be a fruitful direction for future research.

The following section presents a basic generative model that provides a unified mathematical formalism for a large class of linear latent variable models, including PCA (section 1.6.3), FA (section 1.6.2) and ICA (section 1.6.4). This presentation combines elements of [129], [130] and [116]; model fitting is presented using the Expectation Maximization (EM) algorithm [131] to facilitate the unified formalism, but note that each of these algorithms has a direct optimization implementation as well [116]. This framework is useful for understanding the similarities and differences between these ubiquitous models, and also highlights the mathematical assumptions they rely on. Section 1.6.5 then introduces the autoencoder neural network, which retains many of the features of these models while replacing some of those assumptions with ones that offer greater flexibility in model design and optimization.

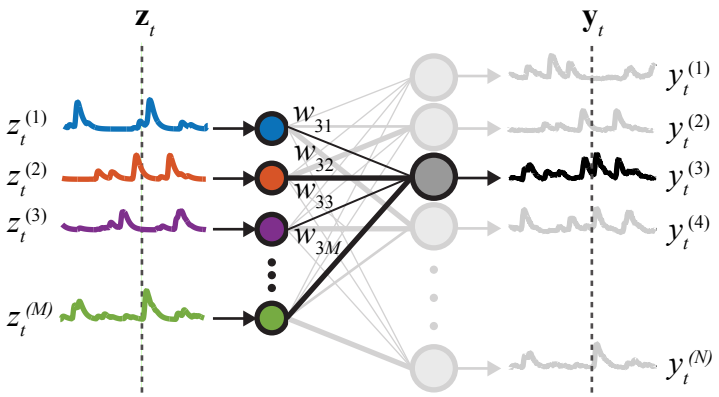

Figure 1.6: **The linear factor model.** An illustration of the linear factor model, in which the observed activity $\mathbf{y}_t$ is a linear combination of latent variables $\mathbf{z}_t$, corrupted by noise (not shown). Different linear factor models like PCA, FA and ICA make different assumptions about the form of the noise and the prior distribution over the latent variables.

1.6.1 Linear factor model framework

Relating latent variables to neural responses. The basic generative model defines a linear transformation from unobserved latent variables $\mathbf{z}_t \in \mathbb{R}^M$ to observed neural responses $\mathbf{y}_t \in \mathbb{R}^N$, and both signals are corrupted by additive Gaussian noise (figure 1.6):

$$\mathbf{z}_t = \mathbf{d}_t \qquad \mathbf{d}_t \sim \mathcal{N}(0, Q) \tag{1.21}$$

$$\mathbf{y}_t = W\mathbf{z}_t + \mathbf{e}_t \qquad \mathbf{e}_t \sim \mathcal{N}(0, R) \tag{1.22}$$

where $W \in \mathbb{R}^{N \times M}$ is the observation matrix, $\mathbf{d}_t$ and $\mathbf{e}_t$ are noise terms in the latent space and neural space respectively, and $\mathbf{z}_t$ are the latent variables[2]. For simplicity of exposition, the mean of the observed data $\mathbf{y}_t$ is assumed to be zero. The noise

[2] A more general expression for the latent variables that defines a linear dynamical system would be $\mathbf{z}_t = A\mathbf{z}_{t-1} + \mathbf{d}_t$, where A is a state transition matrix. This term is omitted from equation 1.21 because none of the models considered here include A.

terms $\mathbf{d}_t$ and $\mathbf{e}_t$ are independent of each other, and each is independent in time. Without loss of generality, the covariance matrix Q can be redefined as the identity I[3].

Under this model, $\mathbf{y}_t$ is a sum of two Gaussians, and so its marginal distribution is given by

$$\mathbf{y}_t \sim \mathcal{N}(0, WW^\intercal + R) \tag{1.23}$$

In order for this model to capture interesting structure in the data, constraints must be imposed on the noise covariance matrix R; without any constraints, a learning algorithm could set $W = 0$ and R to be the sample covariance of the data. The resulting model would thus place all structure in the noise term. Different constraints on R correspond to different models, as discussed in the following sections.

Fitting the model. One approach to fitting the model parameters $\theta = \{W, R\}$ and inferring the latent variables $\mathbf{Z} = \{\mathbf{z}_t\}_{t=1}^T$ is the EM algorithm. This procedure starts with a random initialization for θ and uses this to infer the latent variables $\mathbf{Z}$ (E-step); then, given $\mathbf{Z}$, θ is updated (M-step). The EM algorithm alternates between these two steps until a predefined convergence criterion is met. Note that the EM algorithm is guaranteed to increase the likelihood on each iteration, but is not guaranteed to find the global optimum of the likelihood function [131].

E-step. Given $R^{(k)}$ and $W^{(k)}$, the parameter estimates from iteration k, the latent variables on iteration $k+1$ can be inferred by using Bayes Rule to calculate

[3]Since Q is a covariance matrix, it is postive definite and can be diagonalized as $Q = VDV^\intercal$. Then, an equivalent model can be defined as $\mathbf{y}_t = WVD^{1/2}\mathbf{z}'_t + \mathbf{e}_t$, where $\mathbf{z}'_t = \mathbf{d}'_t$ and $\mathbf{d}'_t \sim \mathcal{N}(0, I)$.

the posterior distribution:

$$p(\mathbf{z}_t|\mathbf{y}_t) = \frac{p(\mathbf{y}_t|\mathbf{z}_t)p(\mathbf{z}_t)}{p(\mathbf{y}_t)} \tag{1.24}$$

$$= \frac{\mathcal{N}(W^{(k)}\mathbf{z}_t, R^{(k)})\mathcal{N}(0, I)}{\mathcal{N}(0, W^{(k)}(W^{(k)})^\intercal + R^{(k)})} \tag{1.25}$$

$$= \mathcal{N}(\beta^{(k)}\mathbf{y}_t, V^{(k)}) \tag{1.26}$$

where $\beta^{(k)} = (W^{(k)})^\intercal[W^{(k)}(W^{(k)})^\intercal + R^{(k)}]^{-1}$, $V^{(k)} = I - \beta^{(k)}W^{(k)}$, and the substitutions from the first to the second line follow from the model definition (equations 1.21, 1.22 and 1.23). The estimate of the latent variables on iteration $k+1$ is then given by their expected value, $\hat{\mathbf{z}}_t^{(k+1)} = \beta^{(k)}\mathbf{y}_t$. Note that because of the lack of temporal dependence between the $\mathbf{z}_t$, this can be performed for each time point independently.

M-step. Estimation of W can be performed using maximum likelihood estimation given the full set of inferred latent variables $\hat{\mathbf{Z}}^{(k+1)} \in \mathbb{R}^{T\times M}$, their estimated covariance $V^{(k)}$, and the data $\mathbf{Y} \in \mathbb{R}^{T\times N}$:

$$W^{(k+1)} = \mathbf{Y}\left[\hat{\mathbf{Z}}^{(k+1)}\right]^\intercal \left(\hat{\mathbf{Z}}^{(k+1)}\left[\hat{\mathbf{Z}}^{(k+1)}\right]^\intercal + TV^{(k)}\right)^{-1} \tag{1.27}$$

The estimation of R depends on the specific model, but for completeness the unconstrained maximum likelihood estimate is

$$\hat{R}^{(k+1)} = \hat{S} - \frac{W^{(k+1)}\hat{\mathbf{Z}}^{(k+1)}\mathbf{Y}^\intercal}{T} \tag{1.28}$$

where $\hat{S}$ is defined to be the sample covariance matrix of the data $\mathbf{Y}$. The following sections will define how this estimate is modified, depending on the model.

1.6.2 Factor Analysis

The model defined in equations 1.21 and 1.22 becomes maximum likelihood Factor Analysis (FA) when R is constrained to be diagonal. This constraint can be easily implemented by calculating the full, unconstrained maximum likelihood estimate $\hat{R}$ (equation 1.28) and then setting the off-diagonal elements to zero [129].

With this choice of constraint, the observation matrix W (known as the *factor loading matrix* in FA terminology) captures the correlational structure among the dimensions of $\mathbf{y}_t$, while the diagonal of R captures the variance that is unique to each dimension. For this reason FA has been popular in the neuroscience literature, because the variance of each neuron is in general different and depends on its firing rate.

The low-dimensional manifold extracted by FA can be used to discover structure in the high-dimensional data that is difficult to understand at the level of individual neurons. For example, in [121], FA was used to compare the differences in neural variability between spontaneous activity and stimulus-evoked activity. Single-neuron analyses demonstrated that variability decreased with stimulus onset. FA was able to further decompose this variability into a shared variability term (captured by the $WW^\intercal$ term in equation 1.23) and a private variability term (captured by the diagonal R term in equation 1.23), and the authors were able to conclude that stimulus onset caused a much larger decrease in the shared variability term.

FA has also been used to investigate constraints on learning through its use with brain-machine interfaces (BMI). In [123], monkeys learned to control a com-

puter cursor that read neural activity patterns from their primary motor cortex. FA was used to find a low-dimensional manifold that contained most of the neural variability (the *Intrinsic Manifold*, or IM, defined by W in equation 1.23). The mapping from neural activity to cursor movement was then altered by requiring new patterns of activity that fell within the IM or outside of it. Monkeys were able to learn new within-IM mappings quickly, but struggled to learn out-of-IM mappings. This study suggests that the IM is constrained by the structure of the network in which these neurons are embedded, and activity patterns that fall outside of the IM are much more difficult to learn.

One drawback to the use of FA in neuroscience is that it assumes a Gaussian distribution on the noise, whereas the Poisson distribution is often used to describe the discrete nature of spiking activity. A heuristic fix that is often used is to take the square root of the spike counts, which stabilizes the variance and provides a better match between data and model [111], though a Poisson FA has been developed as well [132].

1.6.3 Principal Component Analysis

The model defined in equations 1.21 and 1.22 becomes probabilistic Principal Component Analysis (PPCA) [133] (and the independently defined Sensible Principal Component Analysis [134]) when R is constrained to be a scalar multiple of the identity matrix, $R = \sigma^2 I$. Similar to FA, this constraint can be easily implemented by calculating the full, unconstrained maximum likelihood estimate $\hat{R}$ (equation

1.28) and then setting $\sigma^2 = \text{trace}(\hat{R})/N$ [134]. This choice of constraint does not allow each neuron to have an independent variance, and FA has been shown to significantly outperform PPCA when fitting neural data [111].

The ubiquitous PCA model, though not a proper probability model, can be recovered from the PPCA model by taking $R = \lim_{\sigma^2 \to 0} \sigma^2 I$. The posterior distribution of the latent states collapses to a single point, and the model fitting problem, though still solvable by the EM algorithm, is often performed by diagonalizing the sample covariance matrix $\hat{S} = VDV^\intercal$ and defining the columns of W to be the leading M eigenvectors:

$$\hat{W} = V_M \tag{1.29}$$

$$\hat{\mathbf{z}}_t = \hat{W}^\intercal \mathbf{y}_t \tag{1.30}$$

where V_M denotes the first M columns of the matrix V.

PCA has enjoyed wide use in the neuroscience literature due to its computational simplicity and ease of interpretation. Perhaps its most common use (as in other fields) is as an exploratory data analysis tool, whereby neural activity can be projected into the first two or three PCA dimensions for visualization. For example, in [117], neural activity was recorded throughout larval zebrafish brains during a motor adaptation task. This activity was projected into the first three principal components, and four distinct phases of activity were discovered. These phases were then linked to distinct neural structures whose involvement in the task were previously unknown, and prompted additional experimental work to elucidate the role of these structures in motor learning.

PCA has also been used to study the effect of variability on behavior. In [135], PCA was used to define an "attention" axis in neural activity space by considering average responses across a range of attention conditions in a change-detection task. The location of single-trial activity along this dimension was predictive of the animal's performance in the task, establishing a link between neural variability and behavior. This study was an important first step in understanding the functional role of neural variability, and has spawned numerous studies that continue to utilize similar dimensionality reduction methods [86, 101].

PCA is typically applied to neural activity that has first been averaged over many identical trials and then subsequently smoothed, which destroys the discrete nature of the spike counts. Various extensions to PCA have been developed in the neuroscience literature to address this limitation and others, for instance to more accurately model discrete spike counts (Poisson PCA; [136]), to emphasize the structure associated with particular types of experimental trials (dPCA; [137]), and to emphasize dimensions relevant to dynamical structure (jPCA; [122]).

1.6.4 Independent Component Analysis

ICA, first proposed in the 1980s, was originally formulated to solve the blind source separation problem in the signal processing community [138], and as such much of its terminology and motivation differ substantially from the models already discussed. However, ICA can be interpreted in the above framework and used as a dimensionality reduction tool to supplement PCA and FA [129].

ICA differs from PCA and FA by searching for latent variables that are both *non-Gaussian* and *independent*. Such a solution is attractive in many settings where the Gaussian assumption of FA (and somewhat implicitly PCA) is not appropriate. ICA is a particularly popular preprocessing tool for EEG [139] and fMRI [140], though its use in the analysis of electrophysiology and optophysiology data is not as widespread [124]. I include ICA here because it is a natural model to compare with PCA and FA, which I do in chapter 2 when validating my latent variable framework based on the autoencoder.

A particular version of ICA can be formulated by redefining the linear Gaussian generative model in equations 1.21 and 1.22 to be

$$\mathbf{z}_t = g(\mathbf{d}_t) \qquad \mathbf{d}_t \sim \mathcal{N}(0, Q) \tag{1.31}$$

$$\mathbf{y}_t = W\mathbf{z}_t + \mathbf{e}_t \qquad \mathbf{e}_t \sim \mathcal{N}(0, R) \tag{1.32}$$

where $g(\cdot)$ is defined pointwise as

$$g(x) = \ln\left(\tan\left(\frac{\pi}{4}\left(1 + \operatorname{erf}\left(x/\sqrt{2}\right)\right)\right)\right) \tag{1.33}$$

This choice for $g(\cdot)$ transforms the normally distributed z_t^m to a random variable with the distribution $p(x) = 1/(\pi \cosh(x))$; other choices for $g(\cdot)$ correspond to different prior distributions of $\mathbf{z}$ [129][4]. Under this formulation, ICA can be considered either a linear model with a non-Gaussian prior over the latent variables, or a nonlinear model with a Gaussian prior over the latent variables. Because of this

[4]This technique of transforming latent variables from an easily-sampled distribution to a much more complicated one has recently been explored with great success using variational autoencoders [125] and generative adversarial networks [141], where the form of $g(\cdot)$ is learned from the data.

interpretation, a modified EM algorithm can still infer latent variables and estimate model parameters, but the details are beyond the scope of this section; for more information see [129].

1.6.5 Autoencoders

The mathematical assumption common to the models that fall within the presented framework is that the latent variables are constrained to be independent of one another. This assumption is typically made to increase the tractability of parameter estimation, though has the potential to obscure the true factors underlying neural activity if they are not in fact independent. A related model that does not impose particular distributional assumptions on the latent variables is the autoencoder neural network, which is the basis for all of the models presented in this dissertation.

The *autoencoder* is an unsupervised neural network that takes input data $\mathbf{y}_t$ and tries to copy it to the output $\hat{\mathbf{y}}_t$ (figure 1.7). In the simplest setting, this model is defined as

$$\mathbf{z}_t = W_1\mathbf{y}_t + b_1 \tag{1.34}$$

$$\hat{\mathbf{y}}_t = W_2\mathbf{z}_t + b_2 \tag{1.35}$$

The autoencoder latent variables are a deterministic function of the input data $\mathbf{y}_t$ rather than random variables, and thus this model does not fit perfectly into the generative framework of equations 1.21 and 1.22 (though note that variational autoencoders [125] explicitly define a generative model). However, one interpretation

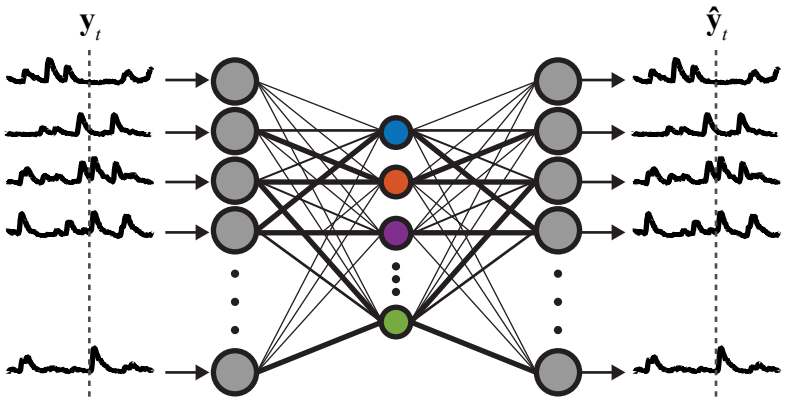

Figure 1.7: **The autoencoder neural network.** An illustration of the autoencoder, in which the observed activity $\mathbf{y}_t$ is mapped to the latent varibles $\mathbf{z}_t$ (represented by the colored nodes), which are then used to construct an estimate $\hat{\mathbf{y}}_t$ of the original input activity $\mathbf{y}_t$. When the mapping from $\mathbf{z}_t$ to $\hat{\mathbf{y}}_t$ is an affine transformation and there are no additional nonlinearities applied to the output, the autoencoder is a linear factor model (note the similarity in structure to figure 1.6).

is that the autoencoder estimates the mean of the posterior distribution of the latent variables, rather than the full distribution [142].

The autoencoder framework is more general than the model presented in equations 1.34 and 1.35: instead of defining the latent variables $\mathbf{z}_t$ to be an affine transformation of the data $\mathbf{y}_t$, and the prediction $\hat{\mathbf{y}}_t$ to be an affine transformation of the latent variables $\mathbf{z}_t$, more general nonlinear functions (called *encoding* and *decoding* functions, respectively) can be used to define a nonlinear latent variable model:

$$\mathbf{z}_t = f_{\text{enc}}(\mathbf{y}_t) \tag{1.36}$$

$$\hat{\mathbf{y}}_t = f_{\text{dec}}(\mathbf{z}_t) \tag{1.37}$$

where f_{enc} and f_{dec} can be any sort of nonlinear function with learnable parameters (i.e. differentiable with respect to those parameters), for example a multi-layer neural network.

Regardless of the functions used for f_{enc} and f_{dec}, model parameters are learned

by minimizing an appropriately-defined reconstruction error $L(\mathbf{y}_t, \hat{\mathbf{y}}_t)$ between the true and predicted activity, such as the mean square error (see section 2.4.1 for a thorough description of the model fitting process). If the autoencoder is not regularized, it is possible to simply learn the identity transformation without capturing interesting features of the data. One approach to preventing this degenerate solution emphasizes learning an overcomplete representation of the input data by allowing more latent variables than input features, and imposing sparsity constraints on the latent variable activations [143]. Another way to prevent the autoencoder from learning the identity transformation is to force an undercomplete representation by only allowing a small number of latent variables, so that the autoencoder becomes a dimensionality reduction technique like PCA, which is the approach used in this dissertation.

The autoencoder is of practical interest to the research presented in this dissertation for several reasons, all of which stem from the use of unconstrained, direct optimization techniques to simultaneously infer latent variables and estimate model parameters. The ability to use a black box numerical optimizer permits a wide range of extensions that do not require reformulating the model fitting procedure, including non-negative latent variables (chapter 2), a cost function that is appropriate for Poisson-distributed data (chapter 2), inclusion of multiple layers to implement nonlinear latent variable models (chapters 3 and 4), and additional inputs to condition on stimulus identity (chapters 3 and 4).

Another nice feature of the autoencoder that is derived from the direct optimization approach is the straightforward evaluation of model performance on cross-

validation data: the deterministic mapping from data to latent variables means that once model parameters (network weights and biases) are fit using the training data, cross-validation data can be fed into the model to obtain predicted values. Using the EM approach requires fitting model parameters using the training data as well, but latent variables must be inferred separately using the E-step for the cross-validation data. This additional step tailors the model's predictions to the cross-validation data used, and therefore these models are typically more difficult to objectively evaluate (though see [111] for an interesting solution to this problem).

In addition to these practical considerations, the autoencoder has long been of theoretical interest because it is related to many other popular models in the statistics and machine learning literature. For example, the autoencoder structure in equations 1.34 and 1.35 with M latent variables, trained using mean square error for $L(\mathbf{y}_t, \hat{\mathbf{y}}_t)$, recovers the same low-dimensional subspace as the first M principal components from PCA [144]. Appropriately-defined nonlinearities and additions to the cost function also allow autoencoders to implement FA and mixture of Gaussian models [129]. These relationships have spawned interesting extensions of autoencoders that are not possible with the more rigidly defined models of equations 1.21 and 1.22, including denoising autoencoders [145], contractive autoencoders [146] and variational autoencoders [125], among others. The following chapters continue in this tradition by introducing new extensions to the autoencoder in order to make it an appropriate tool for studying high-dimensional neural data.

Chapter 2: The Rectified Latent Variable Model (RLVM)

2.1 Introduction

The sensory cortex not only represents information from the sensory periphery, but also incorporates input from other sources throughout the brain. In fact, a large fraction of neural activity in the awake sensory cortex cannot be explained by the presented stimulus, and has been related to a diversity of other factors such as stimulation of other sensory modalities [147, 148], location within the environment [149], and numerous aspects associated with "cortical state" [97, 150, 151] including attention [101, 150], reward [152] and state of arousal [153, 154]. Activity in sensory cortex linked to such non-sensory inputs can result in variability in the responses of neurons to identical stimulus presentations, which has been a subject of much recent study [26, 67, 101, 155]. This suggests that a full understanding of sensory cortical function will require the ability to characterize non-sensory inputs to sensory cortex and how they modulate cortical processing.

However, such non-sensory inputs are typically not under direct experimental control nor directly observed, in which case their effects can only be inferred through their impact on observed neural activity. For example, shared but unobserved inputs can lead to noise correlations observable in simultaneously recorded neurons [75,

156], which can serve as a means to predict one neuron's activity from that of other neurons [19, 157, 158]. Noise correlations thus demonstrate one approach to understanding neural variability, and other recent extensions of this idea have used the summed activity of simultaneously recorded neurons [82, 102] and local field potentials [26, 159] to capture the effects of non-sensory inputs. Notably, these approaches all focus on the effects of shared variability on single neuron activity, and thus do not fully leverage the simultaneous recordings from multiple neurons to infer shared sources of input.

An alternative is to jointly characterize the effects of unobserved, non-sensory inputs on a population of simultaneously recorded neurons. This approach is embodied in a class of methods known as latent variable models [160], which aim to explain neural activity over the population of observed neurons using a small number of factors, or "latent variables". Latent variable models evolved from classic dimensionality reduction techniques like Principal Component Analysis (PCA) [117, 161], and encompass a wide range of methods such as Factor Analysis (FA) [121], Independent Component Analysis (ICA) [124], Poisson Principal Component Analysis [136], demixed Principal Component Analysis [137], Locally Linear Embedding [162], Restricted Boltzmann Machines [163], state space models [106, 107, 164], and Gaussian Process Factor Analysis [111, 165, 166].

Here, we propose a new latent variable approach called the Rectified Latent Variable Model (RLVM). This approach leverages two innovations over previous methods. First, it constrains the latent variables to be non-negative (rectified), which is hypothesized to be a fundamental nonlinear property of neural activity [20]

that can lead to important differences in the resulting descriptions of population activity [167]. Indeed, using simulations, we show that rectification is necessary for the RLVM to recover the true activity of non-negative latent variables underlying population activity. The second innovation is that the RLVM avoids several statistical constraints on the latent variables that are necessary in other methods; for example, it does not require them to be uncorrelated (like PCA), independent (like ICA) or follow Gaussian distributions (like FA). To enable such unconstrained estimation of model parameters, we base solutions of the RLVM on an autoencoder [168], which allows the RLVM to efficiently scale up to large datasets from both electrophysiological and optical recordings.

We first describe the RLVM and demonstrate its application it to a synthetic dataset generated to resemble typical large-scale recordings produced by two-photon experiments. This synthetic dataset gives us ground truth with which to compare RLVM performance with a range of other latent variable approaches. We demonstrate that the RLVM outperforms these alternatives across a range of conditions due to the innovations described above. We then apply the RLVM to a large two-photon dataset recorded in mouse barrel cortex during a decision-making task [43]. The relationship between the latent variables inferred by the RLVM and the behavioral observations related to the task revealed that a large proportion of cortical activity is related to non-vibrissal aspects of the behavioral task. Furthermore, consistent with the results on the synthetic dataset, the RLVM had the ability to match or outperform the other tested latent variable approaches, and also identified latent variables most correlated with individual observed aspects of the experiment.

These results were consistent across many neural populations and animals sampled from this dataset, and thus identify consistent types of latent variables governing the diverse set of neurons recorded over many experiments. In total, this demonstrates that the RLVM is a useful tool for inferring latent variables in population recordings, and how it might be used in order to gain significant insights into how and why sensory cortex integrates sensory processing with non-sensory variables.

2.2 Results

2.2.1 Model formulation

The goal of latent variable modeling is to describe the activity of many simultaneously recorded neurons with a small number of latent variables. Consider a population of N neurons, with the ensemble of observed activity at time t represented by a vector $\mathbf{y}_t$; this observed activity could, for example, be spike counts from multi-electrode recordings or fluorescence values from two-photon microscopy. The M latent variables will also have a vector of activity $\mathbf{z}_t$ at each time point, where M is a free parameter of the model (figure 2.1). The RLVM then attempts to predict the population activity $\mathbf{y}_t$ as a function $f(\cdot)$ of a linear combination of the latent variables $\mathbf{z}_t$

$$\hat{\mathbf{y}}_t = f(W\mathbf{z}_t + \mathbf{b}) \tag{2.1}$$

where W is a matrix of weights that describes how each neuron is coupled to each latent variable and b is a vector of bias terms that account for baseline activity. For two-photon data, it is appropriate to use a linear function for $f(\cdot)$, while for spiking

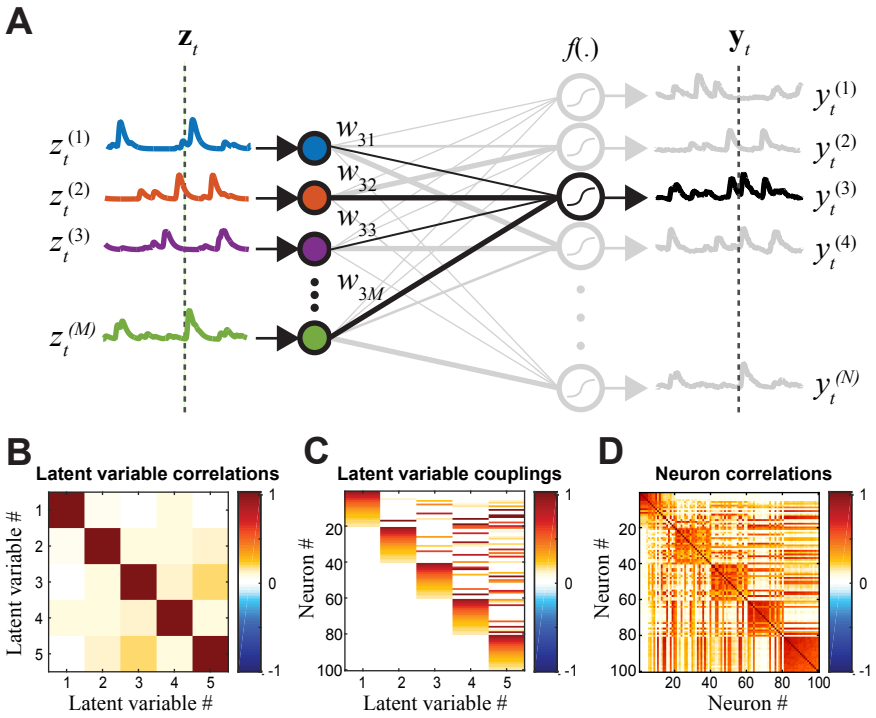

Figure 2.1: **RLVM structure.** **A**: the RLVM predicts the observed population response $\mathbf{y}_t = [y_t^{(1)}\ y_t^{(2)} \dots y_t^{(N)}]^\intercal$ at a given time point t (dashed line, *right*) using a smaller number of non-negative latent variables $\mathbf{z}_t = [z_t^{(1)}\ z_t^{(2)} \dots z_t^{(M)}]^\intercal$ (dashed line, *left*). The latent variables are weighted by a matrix W such that w_{ij} is the weight between neuron i and latent variable j, and the resulting weighted inputs are summed and passed through a nonlinearity $f(\cdot)$. There are additional offset terms for each neuron, not pictured here. **B**–**D**: the hypothesized structure of the cortical network motivating the RLVM formulation is used to generate synthetic data, using 5 latent variables. **B**: factors underlying cortical activity will often be correlated with each other, and our simulation of cortical activity used the correlation matrix shown between latent variables in generating simulated activity. **C**: the weight matrix between latent variables and each neuron, generated to approximate the coupling matrices found with experimental data (compare to figure 2.4B). **D**: the measured pairwise correlation matrix between neurons, computed from simulated data. The correlations predicted by the RLVM arise solely from shared latent variable input and their correlations with each other, rather than pairwise coupling.

data one can use a function that results in non-negative predicted values to match the non-negative spike count values [18].

The vector of latent variables $\mathbf{z}_t$ will in principle represent all factors that drive neural activity, including both stimulus-locked and non-stimulus-locked sig-

nals. These factors may or may not be related to observable quantities in the experiment. For example, they could be related to "external" observables like motor output [122] and pupil dilation [169] or "internal" observables like the local field potential [26], population rate [82], or amount of dopamine release [170]. However, while latent variables might be related to experimental observables, here we make no assumptions on such relationships in determining them.

The non-negative assumption on these latent variables is a key distinction between the RLVM and other latent variable models. This assumption is motivated by the non-negativity of neuronal firing rates and spike counts, which presumably underlie the sources of input being represented by the latent variables. This is not to imply that neural activity cannot represent negative variables; for example, neurons that have high spontaneous firing rates can represent negative quantities as a decrease below that baseline rate. The RLVM does in fact allow for this situation when a latent variable has a nonzero baseline value, because it can have positive and negative deviations from that baseline. Furthermore, the extent to which a latent variable takes advantage of the rectification is learned by the model and does not need to be specified a priori. Thus, by explicitly incorporating rectification, the RLVM finds solutions that are not easily generated by other approaches.

Indeed, rectified latent variables will often be zero, and thus also generate sparse responses, which serves as a second motivation for the nonnegativity of the latent variables. Sparse processes could, for example, represent episodic inputs into cortex from the environment or from other cortical areas. Many models of neural activity cannot in principle find sparse latent variables: for example, they will rarely

explain large amounts of variance (PCA) and will never have a Gaussian distribution (FA). One of the advantages of the RLVM is that even though it does not require distributional assumptions, it is in fact able to capture sparse latent variables (see, for example, figure 2.5A, *latent variable 3*), since rectification can force many values to be zero.

The nonnegative assumption also addresses the "rotational degeneracy" characteristic of any model that contains a matrix-vector multiplication, as in equation 2.1. In such cases, there is no unique solution because, for any orthogonal matrix U, the two solutions $W\mathbf{z}_t$ and $(WU^\intercal)(U\mathbf{z}_t)$ are equivalent since $U^\intercal U = I$ (the identity matrix). To address this issue, different latent variable models considered in this report impose different constraints. For example, PCA assumes that the latent variables are uncorrelated and explain the greatest amount of variance, FA assumes that the latent variables are independent and normally distributed, and ICA assumes that the latent variables are as independent as possible. The RLVM employs a different constraint - nonnegativity of the latent variables - that we expect to be more faithful to the true constraints of neural processing, as described above.

A second key innovation of the RLVM is how it is fit to datasets. We initially used the MML algorithm, which is similar to the EM algorithm for fitting latent variable models [107]. However, inferring the time course of latent variables is challenging because of their high dimensionality, as they have a different value for each time point in the experiment. Fitting the model using random initializations for both the latent variables $\mathbf{z}_t$ and model parameters $\{W, \mathbf{b}\}$ is unlikely to find the best solutions given such a high-dimensional space. As a result, we used an autoen-

coder framework [168] to fit all model components simultaneously. The autoencoder optimizes both $\mathbf{z}_t$ and $\{W, \mathbf{b}\}$ by minimizing the mean square error (or any appropriate cost function) between the true activity and the activity predicted by 2.15. The resulting autoencoder parameters provide both a reasonable initialization for the MML algorithm as well as a good solution to the RLVM without further fitting (detailed below).

2.2.2 Validation of the RLVM using simulated data

To understand the solutions found by the RLVM relative to other latent variable models, we generated simulated data with five latent variables that provided input to 100 neurons (figure 2.1A). These data were generated under the assumption that the input to each neuron is a weighted combination of a small number of correlated, nonnegative latent variables. The latent variable activity was filtered by each neurons coupling and passed through a spiking nonlinearity to produce its firing rate, which was then used to randomly generate spike counts with a Poisson process. Because in this study we are considering application to two-photon imaging data, we then further processed each neurons activity by convolving the generated spike trains with a kernel to simulate calcium dynamics and finally adding Gaussian noise. It should be noted that while this method of generating neural population activity reflects the mathematical form of the RLVM, both this simulation and the form of the RLVM are designed to describe the types of latent variables driving real neural data (as motivated above).

Evaluation of RLVM fitting methods. We first consider the RLVM applied to these simulated data. The RLVM is fit in two stages. In the first stage, an autoencoder is fit to the data and efficiently converges to solutions for the latent variables and parameters. In the second stage, the MML algorithm is initialized with the autoencoder solutions and can then explore solutions to the latent variables that can have more general forms because of the less restrictive constraints. Here we compare the quality of the model fits resulting from this two-stage procedure (autoencoder initialization) to those resulting from a random initialization of the MML algorithm. To quantify the goodness of fit for each model type (random initialization versus autoencoder initialization) we calculated the Pearson correlation coefficients (r) between the true and inferred latent variables. Using random initializations led to poor solutions for the latent variables ($r = 0.781 \pm 0.020$; mean $r\pm$SE over 20 initializations), whereas the two-stage procedure led to far more accurate solutions ($r = 0.971 \pm 0.001$). The superior results achieved by initializing with the autoencoder solution (itself initialized randomly) are due to the high dimensionality of the problem - in the MML algorithm employed here there are relatively few constraints imposed on the latent variables (nonnegativity and some degree of smoothness), which results in many local minima. In contrast, the latent variables of the autoencoder are constrained to be a linear combination of the recorded population activity, and this constraint results in a much smaller space of model solutions.

In fact, we found that the latent variables resulting from the two-stage procedure were extremely similar to the initial values found by the autoencoder itself ($r = 0.994 \pm 0.000$). The main difference between these solutions is that the MML

optimization, which starts from the autoencoder solutions, smooths the time course of the latent variables, whereas the autoencoder latent variables are not generally smooth. As a result, except where otherwise stated, we use the autoencoder solution - forgoing the MML step of the algorithm - as a proxy for the full RLVM performance. Because the latent variables from the autoencoder do not have to be separately inferred for cross-validation data, this choice also provides a more direct comparison of the RLVM with the other latent variable models considered below.

We also tested how the performance of the RLVM depends on parameters governing both the simulated data generation and the fitting procedure to better understand the autoencoders sensitivity to these variables. We found that the autoencoder can accurately recover the latent variables and coupling matrix even with small amounts of data (figure 2.8A) and low SNR (figure 2.8B). We explored the sensitivity of the autoencoder to different values of the regularization parameter on the encoding and decoding weights (λ_1 and λ_2, respectively, in equation 2.18), and found that the results obtained by the autoencoder are constant across several orders of magnitude (figure 2.8C). In practice, we also found that the autoencoder solutions were robust given random initializations of the autoencoder parameters, suggesting that the model was not prone to local minima. These experiments suggest that the autoencoder is a robust fitting method for the RLVM that does not need large amounts of data or precise tuning of optimization parameters to produce accurate results.

We also tested whether the RLVMs nonnegativity constraint is essential for recovering the correct latent variables from the simulated data. Again using r as

a goodness-of-fit measure for the inferred latent variables, we fit the RLVM to the simulated data (using the autoencoder) with different functions for $g(\cdot)$ in equation 2.14. We found that using the rectified nonlinearity (ReLU function) led to much more accurate solutions ($r = 0.963 \pm 0.002$; mean $r \pm$SE over 20 initializations) than using a non-rectified (linear) version of the RLVM ($r = 0.573 \pm 0021$). The linear version places no constraints on either the latent variables or the coupling matrix and thus cannot resolve the rotational degeneracy described above, which results in infinitely many ways for this linear model to reconstruct the observed activity [i.e., the two solutions $W\mathbf{z}_t$ and $(WU^\intercal)(U\mathbf{z}_t)$ are equivalent for any orthogonal matrix U]. Although this implies that the linear version could in principle find the correct non-negative latent variables, the lack of constraints makes it unlikely that the linear version will identify the true latent variables. This illustrates the importance of using the nonlinearity to enforce the nonnegativity of latent variables, in order for the RLVM to recover the latent variables generated with such a nonnegative constraint.

Comparison to other latent variable models. To understand how the RLVM compares with other latent variable methods, we also fit PCA, FA, and ICA models to the simulated data (figure 2.2). We first compared the latent variables inferred by the different models (figure 2.2A), using a measure, maxcorr, that identifies the maximum correlation between each predicted latent variable and the true latent variables (see 2.4.3). The RLVM and FA outperform PCA and especially ICA and are able to largely predict the true latent variable activity once the number of inferred latent variables matches the true number of latent variables. An important

feature of the RLVM and FA fits is that these two methods still infer meaningful latent variables even when the number of inferred latent variables is incorrectly specified: when the number of inferred latent variables is larger than the true number, both methods infer one latent variable that is highly correlated with each of the true latent variables and the remaining inferred latent variables just capture noise in the data.

Because of this behavior, the performance of the RLVM and FA with respect to the maxcorr measure does not decline. The good performance of the RLVM was expected, given that the data were generated according to the assumptions of that model. The good performance of FA in reproducing the latent variables was somewhat surprising, given that it assumes that the latent variables are independent Gaussian variables. However, this assumption only applies in determining the initial coupling matrix, and the FA performance results from how the final coupling matrix is determined through varimax rotation (MATLAB default). The varimax rotation criterion maximizes the variance of the squared entries in each column of the coupling matrix, summed across all columns [171]. This has the effect of changing the weights in each column so that only a few weights are of large magnitude, while the rest are close to zero. Because the true coupling matrix mostly has this structure, FA is able to accurately capture that structure by varimax rotation. Once this final coupling matrix has been found, the resulting latent variables are then determined by linear regression (MATLAB default), which makes no assump- tions about their distribution. PCA and ICA do not infer the correct latent variables because they make assumptions about the latent variables being uncorrelated (PCA)

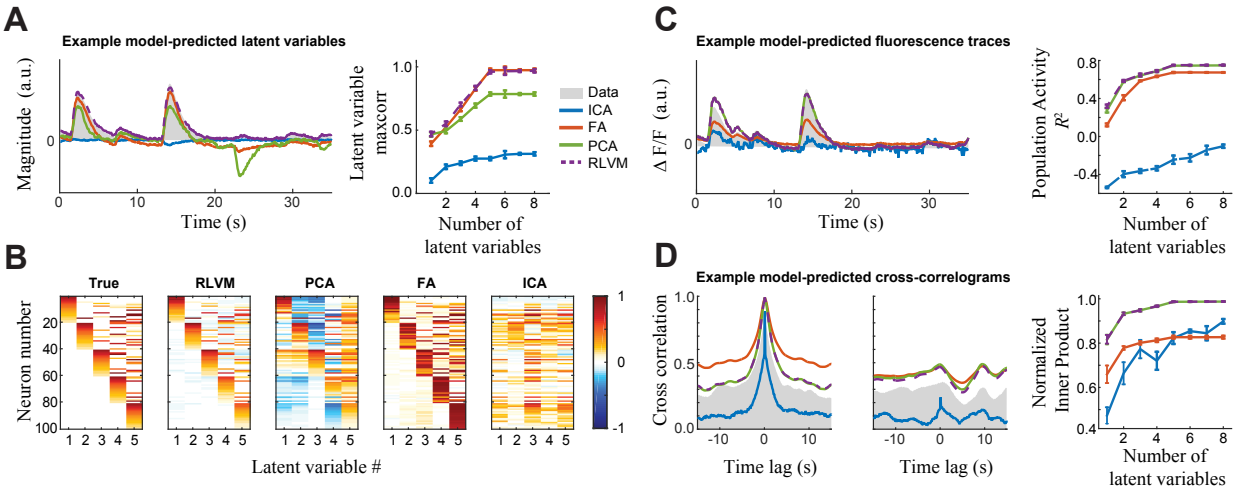

Figure 2.2: **Comparisons between latent variable methods applied to simulated data.** Four different latent variable methods were fit to data that were simulated with 5 latent variables (figure 2.1, BD) and evaluated with cross-validated model performance measures. All error bars represent the SE over cross-validation folds. A, C, and D, *left*, demonstrate results from models with the correct number of latent variables, but performance measures (right) explore different numbers of latent variables. **A**, *left*: time course of a representative latent variable compared with the equivalent inferred latent variable from each method. Note that the FA and RLVM methods are both highly overlapping with the true latent variable. a.u., Arbitrary units. *Right*: latent variable maxcorr, which measures the correlation between the true and inferred latent variables, plotted against the number of latent variables specified during the fitting procedure. Model performance in each case plateaus for the true number of latent variables, indicating that even when overspecifying the number of latent variables the RLVM and FA still infer latent variables that match the true ones. **B**: matrices of coupling weights between neurons and latent variables, inferred by each method. For comparison, the coupling matrix used to generate the simulation is shown on left (reproducing figure 2.1C). **C**, *left*: representative simulated fluorescence trace of one neuron compared with the corresponding trace predicted by each method. Despite their performance in predicting the latent variables (A), here FA does poorly and PCA does well, as does the RLVM. *Right*: R^2 values (median across neurons) between true and predicted fluorescence traces. **D**, *left*: cross-correlograms between two example pairs of simulated neurons compared with the corresponding cross-correlograms based on traces predicted by each method. *Right*: ability of each method to reproduce the pairwise cross-correlations between neurons at zero time lag, measured as the normalized inner product between the true correlation matrix and those calculated from predicted traces for each method.

or independent (ICA), neither of which is true of the simulated data.

A second aspect of the performance of the different latent variable models was based on how well each method captured the coupling weights between these latent variables and each neuron. In this regard, the RLVM and FA performed much better than PCA or ICA (figure 2.2B). Because the RLVM simultaneously infers the latent variables and estimates the coupling matrix, the accurate inference of the latent variables (figure 2.2A) necessarily implies an accurate estimation of the coupling matrix (assuming the overall population activity is well predicted; see next paragraph). PCA and ICA also estimate both model components simultaneously, but again the strong assumptions these methods place on the latent variables prohibit their accurate estimation of the coupling matrices. For FA the initial coupling matrix resembled that of PCA, but the final coupling matrix resulting from varimax rotation bears a much closer resemblance to the true coupling matrix. However, the varimax rotation was not able to accurately capture gradients in the magnitude of the weights (figure 2.2B; compare the red diagonal blocks in the FA coupling matrix with the true coupling matrix).

For all four models considered here, the predicted activity of an individual neuron is given by a weighted sum of the latent variables (figure 2.2A), with weights given by the proper row of the coupling matrix (figure 2.2B). To quantify the accuracies of the resulting model predictions, we used the coefficient of determination (R^2; see section 2.4.3) between the true and predicted activity (figure 2.2C). Interestingly, even though the RLVM and FA produced similar latent variables and coupling matrices, FA did not predict the population activity as well as the RLVM.

This was mostly due to many large weights in the FA coupling matrix, which result from the varimax rotation step. Because the final latent variables from the FA algorithm are determined with linear regression by using the varimax-rotated coupling matrix as a predictor for the population activity, the population activity predictions are constrained by an improperly scaled coupling matrix, and the result is that neurons with estimated coupling weights that are too large in magnitude are not well predicted.

Perhaps surprisingly, PCA performed just as well as the RLVM in predicting the observed activity, even though PCA did not infer the correct latent variables or estimate the correct coupling weights. The reason for this is that the RLVM and PCA both minimize the reconstruction error in their cost function (explicitly and implicitly, respectively); however, because PCA does not constrain the latent variables to be positive, it reconstructs the population activity using both positive and negative values. This leads to differences in the latent variables (figure 2.2A, *left*) and coupling matrices (figure 2.2B) but can result in an equivalent prediction of activity (figure 2.2C, *left*). This difference between the RLVM and PCA in their descriptions of the population activity is a crucial point that we return to when evaluating the PCA solutions on real data.

Finally, we evaluated each method on its ability to account for observed correlations between neurons. Many previous approaches have focused on explaining pairwise correlations directly [19, 157, 172], which requires parameters for each pair of neurons. However, as demonstrated by our example simulation, even just five latent variables can produce a complex pattern of pairwise interactions (figure 2.2D,

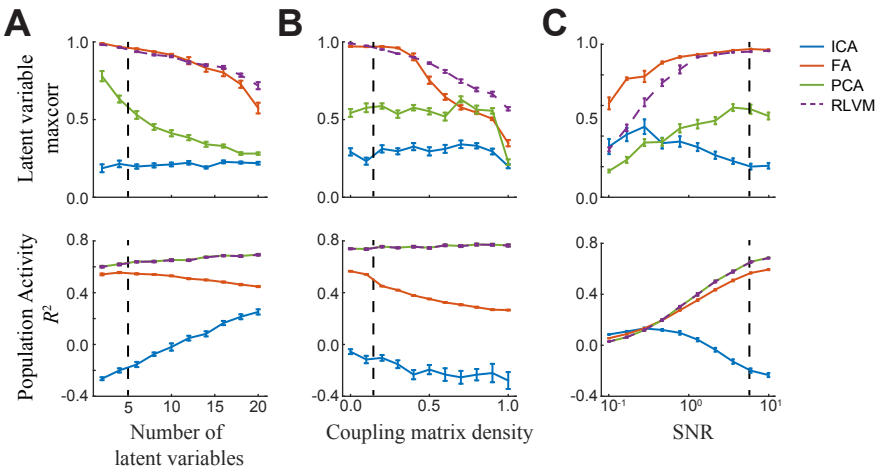

Figure 2.3: **Performance of latent variable methods across a range of simulations.** Simulated data sets are generated as described in figure 2.2, using a range of the number of true latent variables (A), the coupling matrix density (B), and the signal-to-noise ratio (SNR; C). When generating coupling matrices with different numbers of latent variables (A), each method was fit using the true number of latent variables. For the coupling matrices with different densities (B), each neuron had a nonzero weight to at least one latent variable and the coupling matrix density is defined as the proportion of nonzero weights beyond this one-per-neuron baseline. The performance of each method was characterized by the latent variable maxcorr measure (see section 2.4.3) (*top*) and the population activity R^2 (*bottom*). Error bars represent SE over 20 randomly generated data sets. The parameter values used for the simulated data in Fig. 2 are indicated on each plot (*dashed black lines*).

left). Thus latent variable methods offer the ability to explain such correlations using many fewer parameters [158]. To quantify each models ability to capture these correlations, we compare the cross-correlogram at the zero-time-lag point between data and prediction from each neuron pair (which forms the correlation matrix). This agreement was measured using the overlap between the true correlation matrix and the predicted correlation matrix (figure 2.2D, *right*). The results mirror the ability of each method to predict the population activity (figure 2.2C), with the RLVM and PCA capturing more of the correlation structure than FA and ICA.

To demonstrate how the above results generalize to different datasets, we per-

formed a range of simulations while varying the number of latent variables (figure 2.3A), the number of nonzero elements in the coupling matrices (figure 2.3B), and SNR (figure 2.3C). We characterize the performance of each method by its ability to recover the true latent variables (figure 2.3, *top*) and by its ability to reconstruct the population activity (figure 2.3, *bottom*). For all data set variations, the comparison between the RLVM and FA is similar to that seen in figure 2.2: they perform roughly equivalently in their ability to recover the true latent variables, but FA is not able to reconstruct the population activity as well as the RLVM (for the same reasons discussed above). Comparison between the RLVM and PCA reveals the opposite trend: the two perform equivalently in their ability to reconstruct the population activity, but PCA is not able to recover the true latent variables as well as the RLVM (also mirroring the conclusions drawn from figure 2.2).

2.2.3 Application of the RLVM to two-photon experiments

We next applied the RLVM to the experimental data set from Peron et al. (2015) [43]. We selected this data set because it involves a complex task with several "observables" related to behavior and task context, many of which are outside of direct experimental control but potentially related to cortical activity. Additionally, these data included a large number of neurons recorded over long periods of time, which is useful for the performance of any latent variable model. In this experiment, mice performed a pole localization task, in which a pole was lowered at a distal or proximal location next to the animals snout. All but one whisker was trimmed

on that side of the snout, with the single remaining whisker corresponding to the imaged barrel in primary somatosensory cortex (S1). The animal had to signal the location of the pole after a delay period by licking one of two lick ports after the onset of a brief auditory cue. For the analyses in this work, we used a particular subset of available data sets corresponding to different imaged populations of S1 neurons (see table 2.1), selected on the basis of the size of the neural population imaged, the length of time imaged, and its SNR (see section 2.4.5).

For a given imaged population of neurons, we first determined how well the different latent variable methods predicted the observed population activity using different numbers of latent variables (figure 2.4A). The relative performance of the methods was similar to their performance on the simulated data (figure 2.2C, *right*). For the RLVM, PCA, and FA, there was at first a rapid increase in prediction performance as the number of latent variables increased, with the performance beginning to plateau between 5 and 10 latent variables. Because each additional latent variable adds performance, there is no clear number of "true" latent variables that generated the data. However, it is important to note that relatively few are needed before the performance plateaus. Because there is no explicit point where this occurs, we selected a point where there was only a marginal increase in performance (6 latent variables) for all subsequent analyses.

Once the latent variables are determined, the coupling matrix of the RLVM demonstrates how each neuron combines these variables to produce its predicted activity (figure 2.4B). One of the advantages of using two-photon data is that it provides the spatial locations of the neurons, and we can use that information to

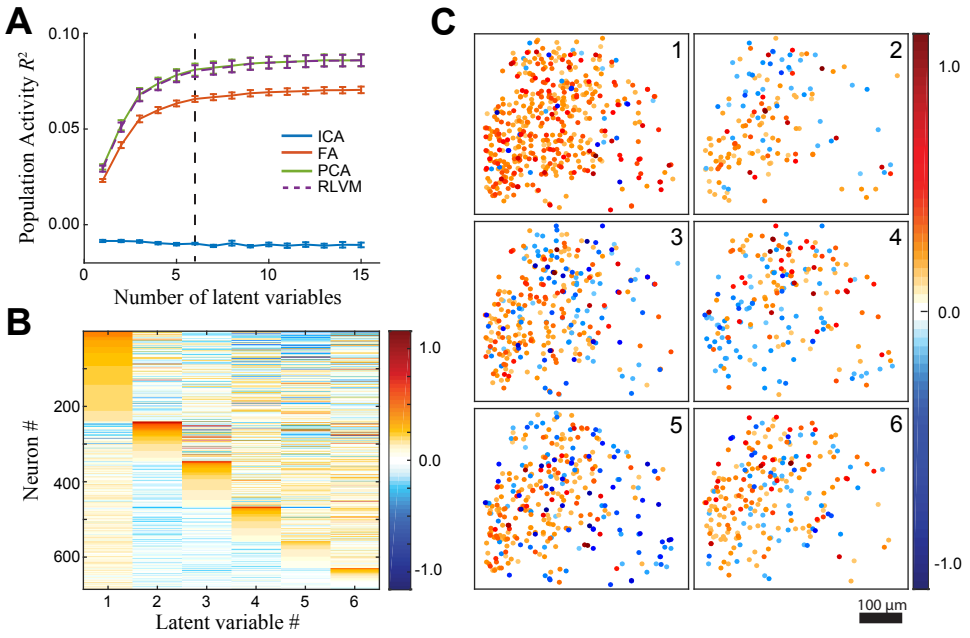

Figure 2.4: **Latent variable methods applied to a two-photon imaging data set recorded in mouse barrel cortex. A**: the performance of each model in reproducing the observed data depends on the number of latent variables used, measured by R^2 between the measured and predicted activity. The relative performance of the different methods is ordered the same as their applications to the simulated data (figure 2.2, *right*). Because there was no clear saturation point of the R^2 values, models with 6 latent variables (dashed black line) were used for subsequent analyses. **B**: the coupling weights of the RLVM between each neuron and each latent variable, with neuron number assigned to aid the visualization of neuron clusters associated with each latent variable (see section 2.4.5). **C**: the spatial positions of neurons coupled to each latent variable. Here neurons whose coupling strength is $> 15\%$ of the maximum coupling strength for the latent variable are shown and color-coded to show the magnitude of their coupling. The imaged neurons were within a single barrel (of mouse primary somatosensory cortex), and the coupling to latent variables exhibited no clear spatial pattern.

determine whether there is any spatial structure in the coupling weights to the latent variables. To look for spatial structure of the neurons coupled to each latent variable, we plotted the spatial locations of the neurons with an absolute magnitude of coupling weight to a given latent variable $> 15\%$ of the maximum absolute magnitude for each (figure 2.4C). The positive and negative weights are intermingled in these plots, and no discernible spatial structure exists. This is expected in part

because these neurons were imaged within a single barrel, and thus all belong to a single cortical column. Nevertheless, this illustrates how latent variables can in principle provide a new way to investigate the functional organization of cortex.

While with simulated data we were able to directly compare the latent variables inferred by each method with the ground truth, the experimental data provide no direct way to validate the latent variables that each method detected. Instead, we hypothesized that latent variables will be related to factors that might be directly observed in the experiment. In this case, there were four "trial variables" measured in this data set: the timing of whisker touches against the pole, the onset of the auditory cue that signals the animal to make its choice, the onset of reward delivery when the animal makes the correct choice, and the timing of licks. We compared the time course of latent variables discovered by the RLVM to these different elements of the experiment, demonstrating clear relationships (figure 2.5A). For example, the activity of latent variable 3 is only active in the same periods where there were whisker touches. In the meantime, both latent variables 1 and 2 and 4 appear to be correlated with the choice cue and/or following reward - note that the animal had high performance, so was rewarded on all trials in this case.

To quantify these relationships, we used linear regression to predict the activity of each latent variable with the four observed trial variables, using R^2 as a goodness-of-fit measure. Linear regression was used in place of a simpler correlation measure because the coefficients for linear regression can include lagged time points, which allowed the regression model to capture the extended temporal response of fluorescence transients (figure 2.5B). A separate linear regression was performed for

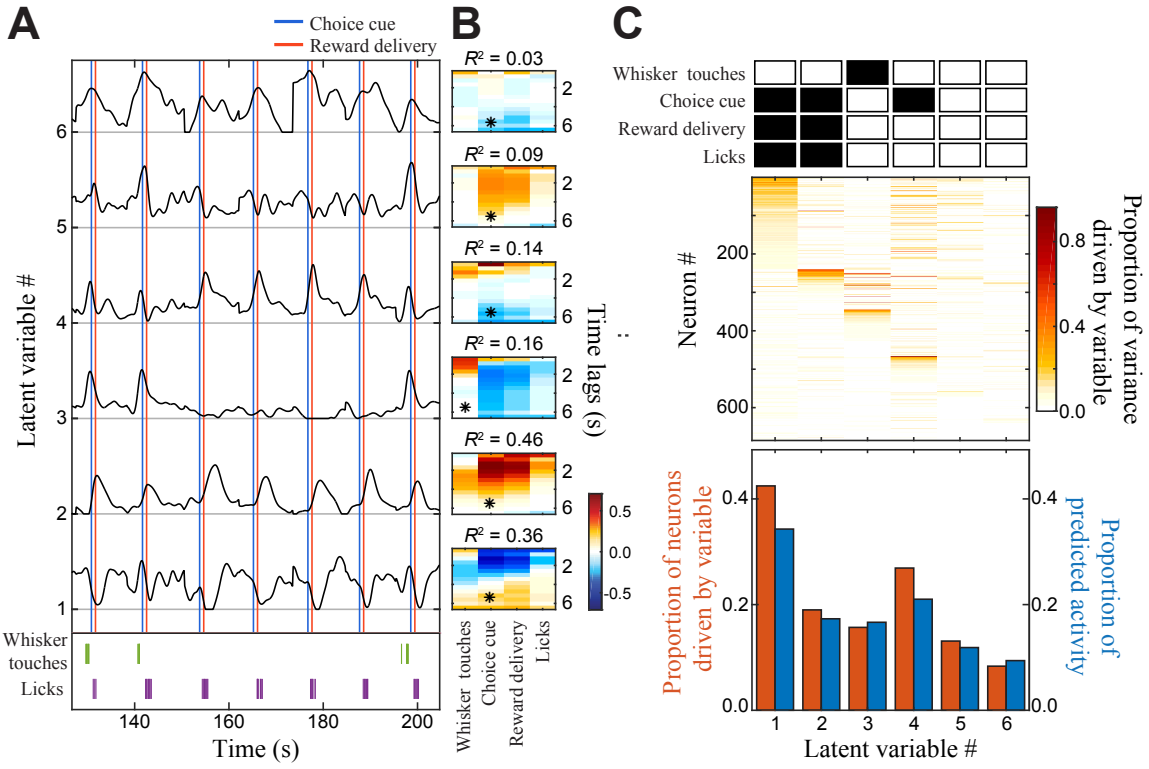

Figure 2.5: **Relationship of latent variables inferred by the RLVM to experimentally observed trial variables. A**: an 80-second sample of the predicted activity of 6 latent variables (extending over 8 task repetitions), demonstrating the relationship between the latent variables and the following "trial variables" observed during the experiment: the auditory cue that signals the animal to make its choice (blue vertical lines), the onset of reward delivery when the animal makes the correct choice (red vertical lines), the timing of whisker touches against the pole (bottom, green), and the timing of licks (bottom, purple). Latent variables are ordered (*bottom* to *top*) based on the magnitude of their variance. **B**: the trial variables at different time lags were used to predict the activity of each latent variable by linear regression, with the relative weights color-coded. The trial variable with the strongest relationship (measured by R^2) is marked with an asterisk, and the corresponding R^2 value is displayed. **C**, *top*: shaded boxes indicate which trial variables are capable of predicting each latent variable. *Middle*: fraction of measured activity of each neuron accounted for by each latent variable. The resulting matrix is related to a weighted version of the coupling matrix (figure 2.4B) and demonstrates the relative contribution of each latent variable to the observed population activity. *Bottom*: fraction of observed neurons driven by each latent variable (red) and relative fraction of predicted neural activity explained by each latent variable (blue).

each trial variable, which did not take into account the correlations that exist among the trial variables. This approach is useful for determining how well latent variables are able to represent single trial variables, as opposed to mixing the influences from multiple trial variables (see below).

The resulting quantitative measures (figure 2.5B) were consistent with the example traces shown: latent variables 1, 2, and 4 are well predicted by the reward portion of the trial, latent variable 3 is well predicted by whisker touches, and latent variables 5 and 6, which do not have any discernible trial-locked patterns, are not well predicted by any of these four trial variables. With these quantitative measures, we can label each latent variable with the set of trial variables that best predict it. To do so, we required that 1) the R^2 value using that trial variable was ¿0.10 and 2) the R^2 value was greater than one-half the largest R^2 value among all trial variables. If both these conditions were met, we considered the latent variable to be "driven" (although perhaps not exclusively) by that trial variable (figure 2.5C, *top*).

Another important question to address was how strongly each latent variable influenced the population response. First, we looked at how strongly a latent variable influenced the population by measuring the proportion of neurons driven by that latent variable. For each neuron we calculated the fraction of the neurons activity explained by each latent variable (figure 2.5C, *middle*; see section 2.4.5) and considered a neuron to be driven by that latent variable if the fraction exceeded 0.10 (figure 2.5C, *bottom*, red bars). We also looked at how each latent variable contributes to the overall proportion of predicted activity, to see if there were any differences between how the latent variables influenced measured versus predicted

responses. We computed a measure similar to that described above but calculated the fraction of the neurons predicted (rather than measured) activity explained by each latent variable (figure 2.5C, *bottom*, blue bars; see 2.4.5). Both measures performed similarly and show that latent variable 1, which was identified with the reward portion of the trial (see above), affected the largest proportion of neurons; latent variable 3, which is the only latent variable identified with the stimulus, affected the third largest proportion; and latent variables 5 and 6, which are not identified with any trial variables, affected the smallest proportions of neurons.

A fundamental feature of the RLVM is its ability to identify "sparsely active" variables, which is enabled by the rectification imposed on latent variable activity. For example, a source of neural activity that is episodically active (such as whisker touches in this case) should mostly have a small magnitude (and explain little variance) except during these events (e.g., latent variable 3 in figure 2.5A). Without rectification, such solutions are challenging for latent variable methods to identify. To demonstrate this, we performed the same analyses as in figure 2.5 using PCA (figure 2.6, A and B). The latent variables inferred by PCA (figure 2.6A) do in fact contain features that are correlated with the trial variables, but these features were more mixed than in the RLVM latent variables. Indeed, latent variable 3 in PCA over the same period does respond to whisker touches but also has activity timed to the choice cue.

A similar observation holds for RLVM latent variables 1 and 2, which are associated with suppressive and excitatory activity during the reward phase, respectively (figure 2.5A). While the RLVM cleanly separates these two subpopulations, they are

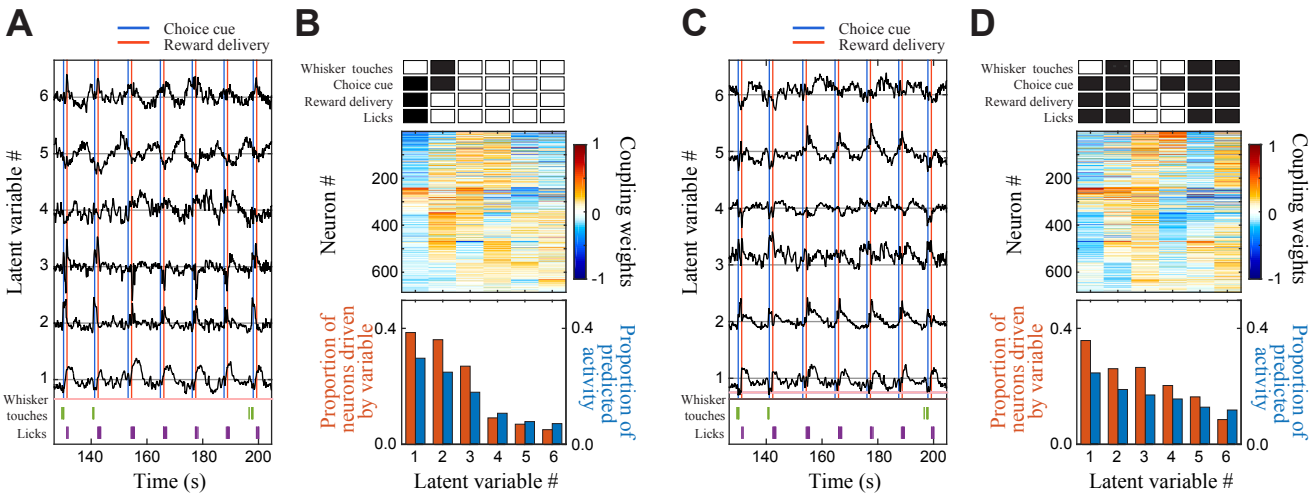

Figure 2.6: **Latent variables inferred by PCA and a linear RLVM show a weaker relationship to individual trial variables.** PCA (A and B) and a linear RLVM (where latent variables were not constrained to be nonnegative) (C and D) were fit to the same experimental data as in figure 2.5. **A**: latent variable time courses inferred by PCA over the same interval as in figure 2.5A, ordered from bottom to top by variance explained. There is a clear mixing of information relative to the RLVM latent variable time courses (figure 2.5A). Latent variable 3, for example, has positive deflections aligned with whisker touches (similar to RLVM latent variable 3) combined with negative deflections aligned with the onset of the reward period (opposite sign relative to RLVM latent variable 4). **B**, *top*: shaded boxes indicate which trial variables are related to the latent variables. *Middle*: coupling matrix between latent variables and each neuron (neurons are ordered the same as those in figure 2.5C). This illustrates how the first few principal components mix inputs from several sources, likely because PCA is based on explaining the greatest fraction of variance with each principal component rather than separating the underlying causes. *Bottom*: summed influence of each latent variable on the population activity (matching measures in figure 2.5C, bottom). **C**: latent variables inferred by a linear RLVM. **D**: same measures as those calculated in B. Both PCA and the linear RLVM latent variables mix features from the RLVM latent variables, which is apparent in their coupling matrices (B and D, *middle*).

mixed together in the first principal component of PCA (figure 2.6B, *middle*; neurons ∼1-100 and ∼250-300, respectively). PCA mixes these two subpopulations because such a combination into a single principal component explains the greatest amount of variance in the data, and this combination is possible because PCA is not restricted to using nonnegative latent variables. These apparent mixtures of latent

variables are also reflected in the coupling matrix between latent variable and neural activity (compare figure 2.6B, *middle*, to figure 2.4B).

To determine whether the nonnegativity constraint on the RLVM is responsible for the differences between the PCA and RLVM solutions, we fit the RLVM on the same data without constraining the latent variables to be nonnegative. The latent variables inferred by this nonrectified version of the RLVM were qualitatively similar to PCAs latent variables (figure 2.6C), and indeed this models latent variables exhibit the same type of mixing as the PCA latent variables. This demonstrates that the RLVMs ability to separate these subpopulations of neurons is mainly due to the rectified nonlinearity and is not just an artifact of PCAs constraint that the latent variables must be uncorrelated.

This example also illustrates that - although the RLVM and PCA are able to explain the same amount of population activity (figure 2.4A) - the underlying latent variables can differ dramatically due to rectification (similar results were seen with FA; data not shown). This same result was seen in the simulated data, with both the comparison between the RLVM and PCA (figure 2.2A, *left*) and the comparison between the RLVM and nonrectified version of the RLVM. This suggests that - if population activity is indeed composed of nonnegative latent variables - the structure of the RLVM makes it a more appropriate method for studying neural population activity.

To demonstrate that the above results from the RLVM (figure 2.4 and figure 2.5) are consistent across different populations of neurons and different animals, we repeated these analyses using nine different populations of neurons from the S1

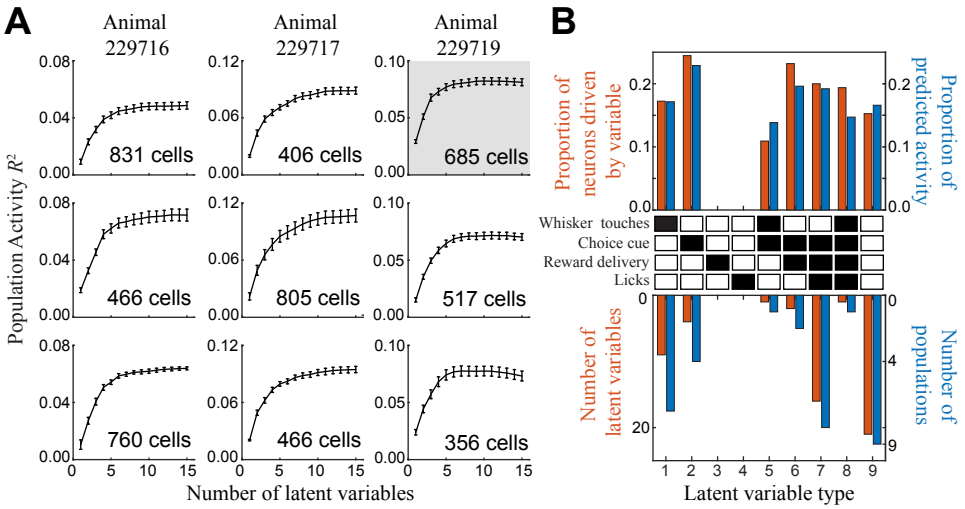

Figure 2.7: **Consistent classifications of latent variables detected across experiments. A**: R^2 between the measured activity and the activity predicted by the RLVM for different imaged populations of neurons. Highlighted plot corresponds to the population of neurons analyzed in figures 2.4-2.6 and reproduces the RLVM values in figure 2.4A. Across experiments there is a similar dependence of R^2 on the number of latent variables, although the overall magnitude of R^2 values depends on the number of neurons and the noise level of each experiment. **B**, *top*: amount of variability accounted for by each "type" of latent variable across all nine populations, using six latent variables per population (same measures as calculated in figure 2.5C, averaged across the number of latent variables of each type). Even though all imaged populations were located in primary somatosensory cortex, across experiments most of the neural activity was related to nontactile sources. *Middle*: latent variables are classified by the combination of trial variables each is related to (same criteria as used in figure 2.5C). *Bottom*: red bars indicate the total number of latent variables in each class (out of 54 total latent variables), and blue bars indicate the total number of populations that contain at least one example of the latent variable class (out of nine total populations). Latent variable types identified with whisker touches (1, 5, and 8) comprise a smaller proportion of the latent variables than types identified with the reward portion of the trial (2 4, 6, and 7). Latent variables that were not identified with any trial variables (9) were present in every population and had an influence on the population activity comparable to the other latent variables.

data set (see table 2.1 for more detailed information). The nine populations contain anywhere from 356 to 831 neurons, and the prediction performance of the RLVM for each population is plotted in figure 2.7A. It is interesting to note that all of these curves mostly plateau before reaching 10 latent variables, despite the wide range

in the number of neurons in these populations, a result that may be related to the complexity of the behavioral task [173].

To repeat the analyses in figure 2.5, we used six latent variables for each population (figure 2.7B). Values were calculated as before (figure 2.5C) and averaged over latent variables from all nine populations. This meta-analysis shows that the results from figure 2.4 and 2.5 broadly hold across different populations in different animals: the latent variables associated with the reward portion of the trial are found in all but one population and account for the largest proportion of the predicted activity in the populations; latent variables associated with the stimulus are found in the majority of populations; and variables that are not identified with any trial variables are found in all populations. Together, these results (figures 2.4-2.7) demonstrate the usefulness of the RLVM as a tool for studying population responses in cortical areas and suggest that latent variable models will be crucial to arriving at a deeper understanding of cortical function.

2.3 Discussion

Recordings of the activity from large numbers of cortical neurons provide opportunities to gain insight into the underlying factors driving cortical activity. Given that there are fewer variables underlying the activity than the number of neurons being recorded, latent variable approaches provide a way to infer the time course of these underlying factors and their relationship to neural activity. Here we presented the RLVM, which is unique in that it places a constraint on the latent variables

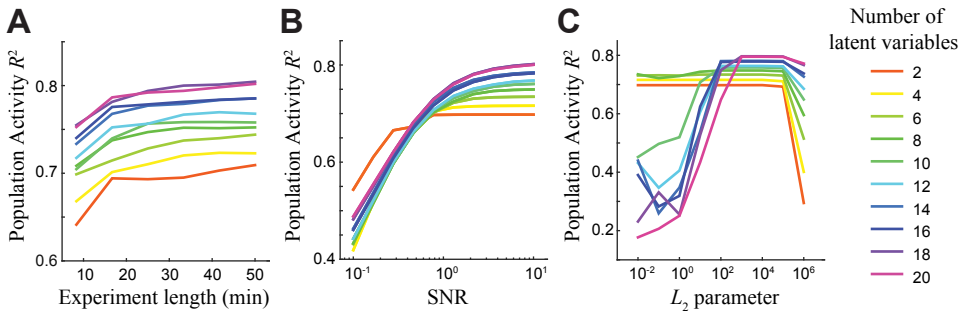

Figure 2.8: **Sensitivity analysis of the autoencoder using simulated data.** Data sets are generated as in figure 2.2 using varying numbers of latent variables. AC: an autoencoder is fit to each data set using the correct number of latent variables. Plotted points represent the mean R^2 value between the true and predicted population activity averaged over 20 such data sets; error bars are omitted for ease of interpretation. Plots show the result of varying the amount of data used for fitting (using 10 Hz sampling rate) (A); the signal-to-noise ratio of the data used for fitting (using 30 minutes of simulated data) (B); or the regularization parameter on the encoding and decoding weight matrices, which were constrained to be equal through weight-tying (again using 30 minutes of simulated data) (C).

that is appropriate for neural activity, namely, that underlying factors are nonnegative (rectified). The RLVM can be fit without relying on a number of statistical assumptions characteristic of past latent variable models, such as the specification of particular distributions for the latent variables. Fitting the RLVM is robust to many aspects of data acquisition and model fitting (figure 2.8) and scales well with increasing numbers of neurons and recording length (figure 2.9).

The results from the simulated data experiments demonstrate that the RLVM is able to recover the true latent variables (figure 2.2A) as well as each neurons coupling weights to those latent variables (figure 2.2B). This guarantees that the method is able to predict single neuron activity well (figure 2.2C) and thus implies that the method is able to accurately capture the structure of the pairwise correlation matrix (figure 2.2D). Importantly, results from the simulated data (figure 2.2

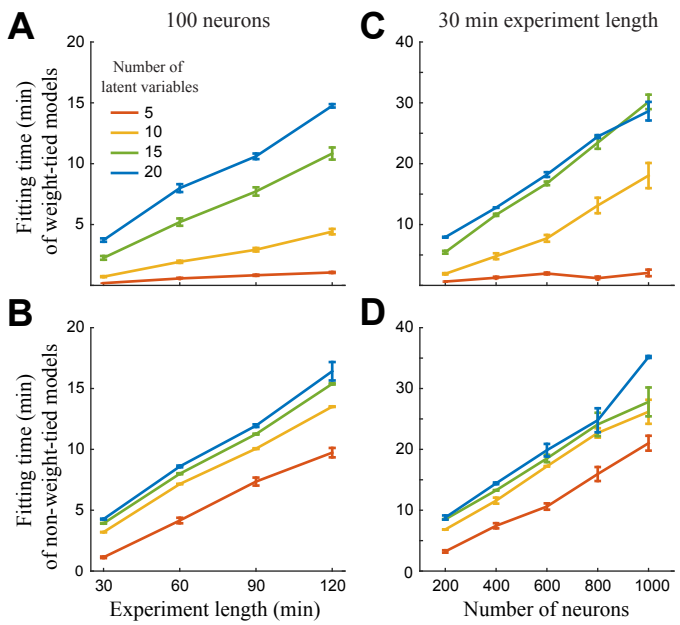

Figure 2.9: **Linear scaling properties of the autoencoder.** A and B: data are generated as in figure 2.2 with 100 neurons and varying recording lengths (with a 10 Hz sampling rate). Autoencoders are fit with and without weight-tying (A and B, respectively). The fitting time scales roughly linearly with the experiment time. C and D: data are generated as in figure 2.2 with a 30 minute experiment time and varying the number of neurons. Autoencoders are fit with and without weight-tying (C and D, respectively). The fitting time scales roughly linearly with the number of neurons. Comparison of A and C (weight-tying) with B and D (no weight-tying) shows that while weight-tying approximately halves the number of estimated parameters, it leads to $>$ 2-fold speedup in fitting time with a small number of latent variables. As the number of latent variables increases this speedup advantage from weight-tying is lost. Plotted values are mean fitting times $\pm$ SE over 20 data sets. These results were obtained on a desktop machine running Ubuntu 14.04 LTS with 16 Intel Xeon E5-2670 processors and 126 GB of RAM; the MATLAB implementation of the autoencoder has not been optimized for this particular architecture.

and figure 2.3) also show how the standard variants of PCA, FA, and ICA can recover erroneous model parameters when fitting nonnegative activity generated from nonnegative latent variables. The manner in which these methods fail is important to consider when using them to analyze and interpret nonnegative data such as two-photon fluorescence traces.

Our results on experimental data demonstrate the utility of the RLVM as a tool

for addressing questions about the structure of joint responses in large neural populations. Some of the latent variables inferred by the RLVM have clear relationships with measured trial variables, suggesting potentially meaningful interpretations of these variables. We also demonstrated that the rectification in the RLVM leads to important distinctions in the description of the population activity compared with a method like PCA, which has consequences for further understanding the role these latent variables play in cortical function.

2.3.1 Relationships to other latent variable models

Latent variable models can be classified into two broad categories: static models, which do not take temporal dynamics of the latent variables into account, and dynamic models, which do. The RLVM has elements of both, although it is more directly comparable to static models like PCA, FA, and ICA. These models are also known as linear factor models, so termed because there is a linear transformation from latent variables to predicted activity. While this need not be the case in the general RLVM framework, the formulation of the RLVM for two-photon data uses this assumption as well [since $f(\cdot)$ in equation 2.2 is linear]. One advantage of the RLVM over these other linear factor models is that the RLVM does not specify any statistical constraints on the latent variables, which allows it to accurately capture correlated latent variables. Furthermore, because of the nonnegativity constraint on the latent variables, the RLVM is able to identify latent variables that more closely resemble the form of expected inputs into the cortex and does not have mul-

tiple equivalent solutions that arise from orthogonal transformations like some linear factor models.

There is a close relationship between the RLVM and PCA. If the nonlinearities $f(\cdot)$ and $g(\cdot)$ in equation 2.13 of the RLVM are linear, and the mean square error cost function is used, then the autoencoder solution of the RLVM lies in the same subspace as the PCA solution [144]. The only difference is that the components of the RLVM can be correlated, whereas PCA requires them to be uncorrelated. However, using nonlinear functions for the activity of the neurons $f(\cdot)$ and/or underlying latent variables $g(\cdot)$ allows the RLVM to capture more complex structure in the data than a linear model like PCA [174].

The RLVM structure also contains elements of dynamic latent variable models, because of its ability to impose constraints on the time course of latent variables via the log-prior term $\log p(\mathbf{Z})$ in 2.3. We used a general smoothing prior when using the full MML algorithm (the results of which are shown in figure 2.5A), which allows latent variable values at time points $t-1$ and $t+1$ to influence the value at time t. This is similar to the smoothing prior of GPFA [111], which allows a latent variable value at time point t to have a more complex dependence on past and future time points. However, as the name implies, GPFA is based on FA and imposes similar statistical constraints on the latent variables that we avoided with the RLVM for reasons mentioned above. Another class of dynamic latent variable models are the state-space models [107], which constrain each latent variable at time t to be a linear combination of all latent variables at time $t-1$. Such models allow the dynamics to be fit to the data directly, whereas the RLVM specifies a fixed

relationship between the time points in the dynamics model. State-space models allow one to model the causal relationship between latent variables but come at the expense of making a strong assumption about the form of that relationship (namely, that latent variables are only determined by their values at the previous time step). Which type of model is most appropriate might then depend on whether dynamics are generated by the latent variables (and/or observed neurons) or by processes extrinsic to the system, such as with the trial variables we considered here. For the applications to the two-photon data set in S1, we found that the solutions for the static and dynamic versions of the RLVM were similar, in part because of the simple dynamics model we imposed. However, the nature of two-photon data does not lend itself to more restrictive dynamics models (like the state-space models) because of the slow timescales. The investigation of more complex dynamics models in the RLVM is a direction for future work.

The analysis performed with the data set from Peron et al. (2015) [43] demonstrates the ability of the RLVM to find latent variables that are correlated with individual task parameters. This "demixing" of task parameters is not an explicit goal of the method but rather results from the rectification of the latent variables (figure 2.5 and 2.6). dPCA [137] is a dimensionality reduction technique that is explicitly formulated to find dimensions that capture variance related to individual task parameters and as such is a mix between supervised and unsupervised dimensionality reduction. The RLVM, in contrast, is a fully unsupervised method, as task parameter information is not used for model fitting. An important restriction of the dPCA method is that it requires neural activity that has been averaged over trial

conditions of the same type, which prevents it from being used to address variability at the level of single trials. Another consequence is that dPCA cannot be used with continuous trial variables (because it requires averaging over similar trial types), and hence we were not able to compare the RLVM to dPCA on the somatosensory data set because of the uncontrolled nature of the stimulus presentation.

The RLVM is thus an unsupervised dimensionality reduction technique and has several advantages as a general method for interpreting population recordings relative to other latent variable approaches, as described throughout this article. However, there are particular situations in which other approaches may yield important insights over the RLVM. If one only wants to estimate the dimensionality of the data or visualize it in two or three dimensions, PCA may be a more appropriate choice. As described in the above paragraph, if one wants to visualize low-dimensional representations of activity that correspond to specific, discrete trial conditions, a targeted dimensionality reduction technique such as dPCA is most appropriate. Such a supervised approach is in contrast to the unsupervised approach we demonstrate here with the S1 data sets, although it is possible to incorporate such conditional dependencies into the RLVM framework, as discussed below.

2.3.2 Model extensions

The RLVM is a flexible model framework because there are many possible extensions that can be incorporated, depending on the desired application area. For example, the RLVM can be fit to spiking data by using a negative log-likelihood

cost function that assumes Poisson noise and specifying $f(\cdot)$ in equation 2.13 (the output nonlinearity) to be a rectifying function (see figure 2.10). This version of the RLVM then becomes comparable against a different set of dimensionality reduction approaches specific to spiking data such as Poisson PCA [136], Poisson FA [175], and Poisson linear dynamical system [108], which also use a rectifying nonlinearity on the model output. However, like the other methods considered in this article, these methods do not place a rectifying nonlinearity on the latent variables, which would still be a defining feature of the RLVM.

One limitation of the RLVM is that it is only able to model additive interactions between the latent variables. Although there is evidence to support the existence of additive interactions in cortex [100], and although they are commonly used for modeling [64, 82, 102], there has been recent interest in modeling multiplicative interactions [67, 96, 101]. It is possible to extend the RLVM to model non-additive interactions by adding more hidden layers to the model. This approach effectively allows a neural network to transform the latent variables into the observed data in a nonlinear manner and is the basis of "stacked" autoencoders [168], which we leave as a direction for future work.

For some analyses, it may not be desirable to have the effects of the stimulus represented in the activity of the latent variables. In this case it is possible to incorporate a stimulus model into the RLVM, such that the activity of each neuron is driven by the weighted sum of the latent variables plus terms that capture the stimulus response. This model formulation would then allow for the investigation of the relationship between stimulus processing and ongoing cortical activity. In a

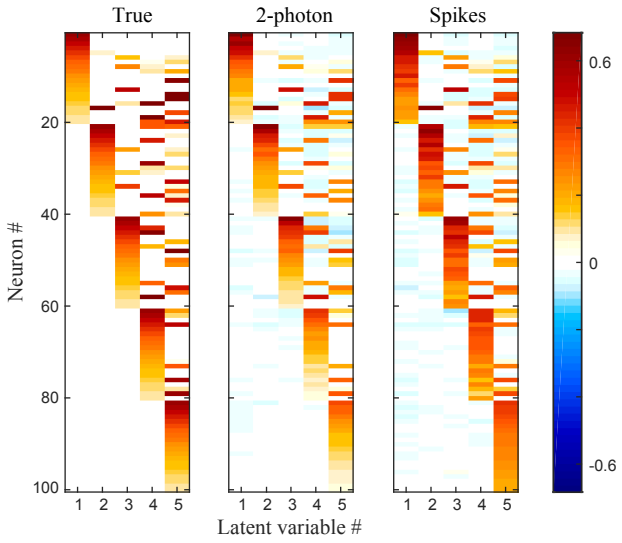

Figure 2.10: **Using the RLVM for spiking data.** *Left*: coupling matrix used to generate synthetic data, as described in section 2.4.4 (reproducing figure 2.1C). *Center*: the estimated coupling matrix when the autoencoder variant of the RLVM is fit to the simulated two-photon data with a Gaussian noise loss function (mean square error). *Right*: estimated coupling matrix when the autoencoder variant of the RLVM is fit to the simulated spiking data using a Poisson noise loss function (negative log-likelihood). The simulated data contained spikes binned at 100-ms resolution. The good agreement of both estimated coupling matrices with the true coupling matrix demonstrates that the RLVM can recover the same model parameters when fit using two different types of data. This result suggests that the RLVM will perform similarly on multi-electrode data, without the need for data smoothing or averaging across trials (which are common preprocessing steps used with spiking data when attempting to use latent variable models not suited for discrete count data, such as PCA [160]).

similar manner, the RLVM can also incorporate additional trial information, so that the inferred latent variables only capture variation in the population response that is independent of this information. Such a model is then more closely related to the dPCA approach discussed above.

The recent development of new recording technologies like high-density multi-electrode arrays and two-photon microscopy is leading to increasingly large and rich neural data sets. We have shown here that the RLVM can be used effectively to

analyze two-photon data sets and that it is also possible to apply this model to spiking data (figure 2.10). The RLVM is thus a simple and extendable model that can be used to analyze both types of large population recordings, and in doing so can help uncover neural mechanisms that may not be obvious when studying the responses of single neurons.

2.4 Methods

2.4.1 Fitting the RLVM

The goal of the RLVM is to accurately predict observed neural activity $\mathbf{y}_t \in \mathbb{R}^N$ using a smaller set of latent variables $\mathbf{z}_t \in \mathbb{R}^M_{\geq 0}$. Here $\mathbf{y}_t$ and $\mathbf{z}_t$ are vectors describing the activity at each time point t, and the matrices $\mathbf{Y} = \{\mathbf{y}_t\}_{t=1}^T$ and $\mathbf{Z} = \{\mathbf{z}_t\}_{t=1}^T$ are all the observed data and latent variables, respectively, across time. The RLVM then tries to predict the observed activity $\mathbf{y}_t$ with the $\mathbf{z}_t$ as follows:

$$\hat{\mathbf{y}}_t = f(W\mathbf{z}_t + \mathbf{b}) \tag{2.2}$$

where $f(\cdot)$ is a parametric nonlinearity and the model parameters are the coupling matrix $W \in \mathbb{R}^{N \times M}$ and the bias vector $\mathbf{b} \in \mathbb{R}^M$, collectively referred to as $\theta = \{W, \mathbf{b}\}$.

Estimation of model components. We estimate the model parameters θ and infer the latent variables $\mathbf{Z}$ using the maximum marginal likelihood (MML) algorithm, following [107] and [158]. The MML algorithm is closely related to the expectation-maximization (EM) algorithm, as both maximize an approximation to

the true log-likelihood function. The MML algorithm first infers the latent variables $\mathbf{Z}$, using initial model parameters $\hat{\theta}^{(0)}$, and then updates the model parameters, using the newly inferred latent variables. Each of these steps corresponds to a *maximum a posteriori* (MAP) estimate of the latent variables and model parameters, respectively, in which we maximize the sum of the data log-likelihood and a log-prior distribution [158]:

$$\hat{\mathbf{Z}}^{(k+1)} = \underset{\mathbf{Z}\geq 0}{\operatorname{argmax}} \ \log p\left(\mathbf{Y}|\mathbf{Z}, \hat{\theta}^{(k)}\right) + \log p\left(\mathbf{Z}\right) \tag{2.3}$$

$$\hat{\theta}^{(k+1)} = \underset{\theta}{\operatorname{argmax}} \ \log p\left(\mathbf{Y}|\hat{\mathbf{Z}}^{(k+1)}, \theta\right) + \log p\left(\theta\right) \tag{2.4}$$

The algorithm continues to alternate between these two steps until a convergence criterion is met. Although equation 2.3 is a constrained optimization problem, it can be transformed into an unconstrained optimization problem as described below, and thus we solve both equations 2.3 and 2.4 with an unconstrained limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method. Like the EM algorithm, the MML algorithm is only guaranteed to find a local, rather than global, optimum. Thus proper initialization (described below) can in principle be important.

The MML algorithm presented in equations 2.3 and 2.4 is a general procedure that can be specifically implemented for different types of data by defining the probability distribution in the data log-likelihood term $\log p(\mathbf{Y}|\mathbf{Z}, \theta)$, which describes the probability of the observations given the current estimates of the latent variables and the model parameters. This term refers to the form of the expected noise distribution when considering what the model predicts versus what is observed. For

example, in what follows we use a Gaussian distribution for two-photon data but could instead use a Poisson distribution for spiking data. The forms of the log-prior terms $\log p(\mathbf{Z})$ and $\log p(\theta)$ are in general independent of the form of the data log-likelihood term. Because this work is focused on the analysis of two-photon data we discuss the implementation of the MML algorithm that is specific to modeling two-photon data, including a discussion of our treatment of the log-prior terms.

We first address the data log-likelihood terms in the form $\log p(\mathbf{Y}|\mathbf{Z}, \theta)$. For two-photon data, we model the observed fluorescence traces as a linear combination of the latent variables plus a bias term, $\hat{\mathbf{y}}_t = W\mathbf{z}_t + \mathbf{b}$ [equation 2.2, with linear $f(\cdot)$]. Furthermore, we assume a Gaussian noise model so that $p(\mathbf{y}_t|\mathbf{z}_t, W, \mathbf{b}) \in \mathcal{N}(\hat{\mathbf{y}}_t, \Sigma)$ and

$$\begin{aligned}\log p(\mathbf{y}_t|\mathbf{z}_t, W, \mathbf{b}) = &-\frac{N}{2}\log 2\pi - \frac{1}{2}\log\det(\Sigma) \\ &- \frac{(\mathbf{y}_t - (W\mathbf{z}_t + \mathbf{b}))^\intercal \Sigma^{-1} (\mathbf{y}_t - (W\mathbf{z}_t + \mathbf{b}))}{2}\end{aligned} \tag{2.5}$$

for a given time point t. The Gaussian noise model captures the measurement noise that corrupts the true fluorescence signal and is commonly used in models of two-photon data [12]. We validated this choice by measuring the distribution of residuals from the RLVM model fits to the experimental data used in figures 2.4-2.6, which are well described by a Gaussian distribution (data not shown). For computational convenience we do not try to fit the noise covariance matrix Σ but rather model it as a constant times the identity matrix. This constant can be incorporated into the log-prior terms and hence does not explicitly show up in the final MML equations (equations 2.10 and 2.11 below). By modeling the noise covariance matrix as a multiple of the identity matrix we are making the assumption

that the Gaussian noise has the same variance for each neuron (isotropic noise). Although not true in general, the advantage of this simplification is that we do not need to estimate the variance parameter, and equations 2.3 and 2.4 become penalized least squares problems when using L_2 regularization, which can be solved analytically. Constraining the noise covariance matrix to be diagonal (anisotropic noise) leads to solving a penalized weighted least squares problem, which must be solved iteratively.

We also make the assumption that data at different time points are conditionally independent, so that the full log-likelihood term can be written as

$$\begin{aligned} \log p\left(\mathbf{Y}|\mathbf{Z},\theta\right) &= \log\left(\Pi_t p\left(\mathbf{y}_t|\mathbf{z}_t,W,\mathbf{b}\right)\right) \\ &= \sum_t \log p\left(\mathbf{y}_t|\mathbf{z}_t,W,\mathbf{b}\right) \\ &= -\frac{1}{2}\sum_t \left\|\mathbf{y}_t - \left(W\mathbf{z}_t+\mathbf{b}\right)\right\|_2^2 + const \end{aligned} \tag{2.6}$$

where $\|\mathbf{x}\|_2^2 = \sum x_i^2$ is the squared L_2 norm of a vector $\mathbf{x}$. The assumption of conditional independence is common practice when dealing with data log-likelihood terms [17] and allows us to factorize the full conditional distribution $\log p(\mathbf{Y}|\mathbf{Z},\theta)$; without this assumption the resulting log-likelihood term would be intractable.

To further constrain the types of solutions found by the model, we choose a particular form of the log-prior term $\log p(\mathbf{Z})$ (equation 2.3). Many different priors are used for $\mathbf{Z}$ in the neuroscience literature on latent variable models, including latent dynamical systems priors [107] and Gaussian process priors [101, 111]. Here we use a simple smoothing prior that penalizes the second derivative of the time course of each latent variable $\mathbf{z}^i$ (where $\mathbf{z}^i$ represents the entire time course of latent

variable i), which can be written as

$$\log p(\mathbf{z}^i) \propto \|D\mathbf{z}^i\|_2^2 \tag{2.7}$$

where D is the discrete Laplace operator,

$$D = \begin{bmatrix} -2 & 1 & & & 0 \\ 1 & -2 & 1 & & \\ & 1 & -2 & \ddots & \\ & & \ddots & \ddots & 1 \\ 0 & & & 1 & -2 \end{bmatrix} \tag{2.8}$$

The matrix-vector multiplication $D\mathbf{z}^i$ computes a discrete version of the second derivative of latent variable i, such that $\|D\mathbf{z}^i\|_2^2$ will be large when $\mathbf{z}^i$ is highly varying (i.e. noisy) and small when $\mathbf{z}^i$ is smooth. The full log-prior term $\log p(\mathbf{Z})$ is the sum of these terms for each individual latent variable.

The log-prior term $\log p(\theta)$ in equation 2.4 likewise allows for the incorporation of additional constraints on model parameters. We use a standard zero-mean Gaussian prior on both the coupling matrix W and the biases $\mathbf{b}$, so that

$$\log p(\theta) \propto \alpha\|W\|_F^2 + \beta\|\mathbf{b}\|_2^2 \tag{2.9}$$

where $\|W\|_F^2 = \sum_{i,j} w_{i,j}^2$ is the Frobenius norm of a matrix W and α and β are constants that scale the relative weight of each term. This prior has the effect of preventing the model parameters from growing too large, which can hurt model performance (see figure 2.3C).

Using the expressions in equations 2.6, 2.7 and 2.9, the two-photon implemen-

tation of the general MML algorithm in equations 2.3 and 2.4 becomes

$$\hat{\mathbf{Z}}^{(k+1)} = \underset{\mathbf{Z}\geq 0}{\operatorname{argmin}} \frac{1}{2}\sum_t \left\|\mathbf{y}_t - \left(W^{(k)}\mathbf{z}_t + \mathbf{b}^{(k)}\right)\right\|_2^2 + \frac{\lambda_z}{2}\sum_i \left\|D\mathbf{z}^i\right\|_2^2 \tag{2.10}$$

$$\hat{\theta}^{(k+1)} = \underset{\theta}{\operatorname{argmin}} \frac{1}{2}\sum_t \left\|\mathbf{y}_t - \left(W\hat{\mathbf{z}}_t^{(k+1)} + \mathbf{b}\right)\right\|_2^2 + \frac{\lambda_W}{2}\left\|W\right\|_F^2 + \frac{\lambda_b}{2}\left\|\mathbf{b}\right\|_2^2 \tag{2.11}$$

The λ values in front of the log-prior terms are hyperparameters that are chose by hand (see section 2.4.2).

The non-negativity constraint on the latent variables $\mathbf{Z}$ is the defining feature of the RLVM. Although it is possible to use explicitly constrained optimization techniques, we take a different approach that is more in line with the autoencoder optimization we use to obtain initial values for the MML algorithm (see below). Instead of optimizing non-negative latent variables $\mathbf{z}^i$, we substitute them with unconstrained latent variables $\mathbf{x}^i$ that are passed through a rectified linear (ReLU) function $g(\cdot)$:

$$z_t^i = g(x_t^i) = \max(0,\ x_t^i) \tag{2.12}$$

The model of neural activity (equation 2.2) then becomes

$$\hat{\mathbf{y}}_t = f\left[Wg\left(\mathbf{x}_t\right) + \mathbf{b}\right] \tag{2.13}$$

where $g(\cdot)$ is applied element-wise to the vector $\mathbf{x}_t$, and unconstrained optimization techniques can be used in equations 2.3 and 2.10 to solve for $\mathbf{X}$ instead of $\mathbf{Z}$. Although the ReLU function is not differentiable at zero, we use the sub-differential approach common in the neural networks literature and define the derivative to be zero at zero [176].

Initialization of model components using autoencoders. In the infer-

ence of $\mathbf{Z}$ (equations 2.3 and 2.10), there are $T \times M$ parameters to estimate (where T is the number of time steps in the experiment and M is the number of inferred latent variables), which is a very high-dimensional space to search for an experiment of reasonable length. The prior distribution we place on the latent variables is not a strong one, and as a result this optimization step tends to get stuck in poor local minima. To avoid poor local minima, we initialize the MML optimization algorithm using initial estimates of both the model parameters and the latent variables from the solution of an autoencoder [144, 168, 174]. An autoencoder is a neural network model that attempts to reconstruct its input using a smaller number of dimensions, and its mathematical formulation is similar to the RLVM - so similar, in fact, that the model parameters in the RLVM have direct analogs in the autoencoder, as shown below. Furthermore, the optimization routine for the autoencoder is faster and better behaved than the MML algorithm, which makes it an attractive model for finding initial RLVM values.

The autoencoder takes the vector of neural activities $\mathbf{y}_t \in \mathbb{R}^N$ and projects it down onto a lower-dimensional space $\mathbb{R}^M$ with an encoding matrix $W_1 \in \mathbb{R}^{M \times N}$. A bias term $\mathbf{b}_1 \in \mathbb{R}^M$ is added to this projected vector, so that the resulting vector $\mathbf{x}_t \in \mathbb{R}^M$ is given by $\mathbf{x}_t = W_1 \mathbf{y}_t + \mathbf{b}_1$. W_1 is said to encode the original vector $\mathbf{y}_t$ in the lower-dimensional space with the vector $\mathbf{x}_t$, which is analogous to the unconstrained latent variables $\mathbf{x}_t$ in the RLVM (equation 2.13). Following the RLVM, we enforce the non-negativity constraint on $\mathbf{x}_t$ by applying the ReLU function:

$$\mathbf{z}_t = g(\mathbf{x}_t) = g(W_1 \mathbf{y}_t + \mathbf{b}_1) \tag{2.14}$$

As with the unconstrained latent variables $\mathbf{x}_t$, there is a direct correspondence between the autoencoder's rectified latent variables $\mathbf{z}_t$ in equation 2.14 and the RLVM's rectified latent variables $\mathbf{z}_t$ in equation 2.2. The autoencoder (again like the RLVM) then reconstructs the original activity $\mathbf{y}_t$ by applying a decoding matrix $W_2 \in \mathbb{R}^{N \times M}$ to $\mathbf{z}_t$ and adding a bias term $\mathbf{b}_2 \in \mathbb{R}^N$. The result is passed through a parametric nonlinearity $f(\cdot)$ so that the reconstructed activity $\hat{\mathbf{y}}_t \in \mathbb{R}^N$ is given by

$$\hat{\mathbf{y}}_t = f\left(W_2 \mathbf{z}_t + \mathbf{b}_2\right) \tag{2.15}$$

which matches the RLVM model structure in equation 2.2. The weight matrices and bias terms, grouped as $\Theta = \{W_1, W_2, \mathbf{b}_1, \mathbf{b}_2\}$, are simultaneously fit by minimizing the reconstruction error $L(\mathbf{Y}, \hat{\mathbf{Y}})$ between the observed activity $\mathbf{Y}$ and the predicted activity $\hat{\mathbf{Y}}$:

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} \; L\left(\mathbf{Y}, \hat{\mathbf{Y}}\right) \tag{2.16}$$

Once this optimization problem has been solved with standard gradient descent methods, we initialize the RLVM model parameters in equation 2.3 with $\hat{\theta}^{(0)} = \{W_2, \mathbf{b}_2\}$. A notable advantage of the autoencoder is that there is no need to alternate between inferring latent variables and estimating model parameters, as in equations 2.3 and 2.4: once the model parameters have been estiamted with equation 2.16, the latent variables can be explicitly calculated with 2.14.

For modeling two-photon data (as above), the noise distribution is Gaussian and the nonlinearity $f(\cdot)$ in equation 2.15 is assumed to be linear. The reconstruction error $L(\mathbf{Y}, \hat{\mathbf{Y}})$ for Gaussian noise is the mean square error (again assuming equal noise variances across neurons), so in this special case of equation 2.16 the

autoencoder estimates for the weights and biases are given by

$$\begin{aligned}\hat{\Theta} &= \underset{\Theta}{\operatorname{argmin}} \frac{1}{2} \sum_t \|\mathbf{y}_t - \hat{\mathbf{y}}_t\|_2^2 \\ &= \underset{\Theta}{\operatorname{argmin}} \frac{1}{2} \sum_t \|\mathbf{y}_t - (W_2 \mathbf{z}_t + \mathbf{b}_2)\|_2^2 \\ &= \underset{\Theta}{\operatorname{argmin}} \frac{1}{2} \sum_t \|\mathbf{y}_t - (W_2 g(W_1 \mathbf{y}_t + \mathbf{b}_1) + \mathbf{b}_2)\|_2^2\end{aligned} \tag{2.17}$$

and we perform this optimization using an L-BFGS routine to obtain the weights and biases.

We also include regularization terms for the model parameters, which prevent overfitting to the training data and can improve the model's ability to generalize to new data [17]. As we saw previously, these regularization terms can also be interpreted as log-prior distributions on the model parameters in the probabilistic setting. A more general optimization problem for the autoencoder that includes both the reconstruction error and these regularization terms is

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} L(\mathbf{Y}, \hat{\mathbf{Y}}) + \frac{\lambda_1}{2}\|W_1\|_F^2 + \frac{\lambda_2}{2}\|W_2\|_F^2 + \frac{\lambda_3}{2}\|\mathbf{b}_1\|_2^2 + \frac{\lambda_4}{2}\|\mathbf{b}_2\|_2^2 \tag{2.18}$$

Large values of λ_i will encourage small values in the corresponding set of parameters. Furthermore, the use of regularization on the weight matrices helps to break a degeneracy in the autoencoder: because the reconstructed activity $\hat{\mathbf{y}}_t$ involves the product between the weights W_2 and the latent variables $\mathbf{z}_t$ (equation 2.15), an equivalent solution is given by the product of $c \times W_2$ and $(1/c) \times \mathbf{z}_t$ for any positive constant c. Applying a regularization penalty to the weights W_2 limits the range of values W_2 can take and thus helps to set a scale for both the weights and the latent variables.

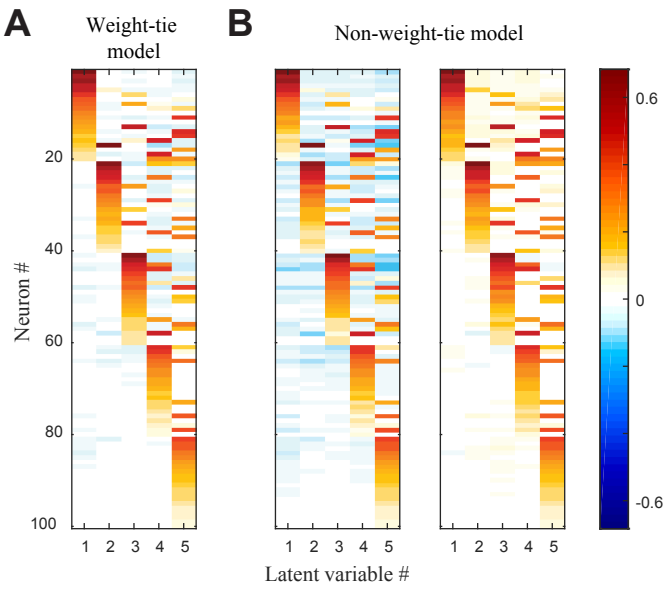

Figure 2.11: **Effect of weight-tying using simulated data**. We compared the effects of weight-tying the autoencoder on the resulting weight matrix by fitting models with and without weight-tying to the simulated data 2.2. **A**: weights learned by the autoencoder when encoding and decoding matrices are constrained to be the same. **B**: encoding (*left*) and decoding (*right*) weights learned by the autoencoder without the weight-tying constraint, demonstrating a pattern very similar to the weight-tied solution in **A**.

We also use "weight-tying" [168] (figure 2.11), where the encoding and decoding weight matrices are constrained to be transposes of each other, i.e. $W_2 = W_1^\intercal$. This has the effect of nearly halving the number of model parameters that need to be estimated, which speeds up the model fitting procedure (figure 2.9), and as a result all models in this chapter initialized with the autoencoder employ weight-tying.

2.4.2 Model fitting details

Fitting the model parameters and latent variable time courses with the MML algorithm requires alternating between inferring latent variables and estimating model parameters. We monitored the log-likelihood values throughout this proce-

dure and ensured that the algorithm stopped only after additional iterations brought no further improvement. We compared the fitting behavior of the MML using random initializations versus autoencoder intializations (section 2.2). For these tests, we used the same regularization values for the latent variables ($\lambda_z = 1$) and for the model parameters (see below) to facilitate model comparisons.

The latent variable models we used to analyze the simulated and experimental data were the RLVM (regularization parameters set as $\lambda_1 = \lambda_2 = 1000/M$, $\lambda_3 = \lambda_4 = 100$, lambdas numbered as in equation 2.18; code available for download at `www.neurotheory.umd.edu/code`), PCA (using MATLAB's built-in function *pca*), FA (using MATLAB's built-in function *factoran*; default settings), and ICA (using FastICA, available for download at `http://research.ics.aalto.fi/ica/fastica/`; default settings). Autoencoder fitting was performed with a MATLAB implementation of the L-BFGS routine by Mark Schmidt [177], available for download at `www.cs.ubc.ca/$\sim$schmidtm/Software/minFunc.html`. The FA results reported with the dataset from Peron et al. 2015 [43] used a PCA-based algorithm (available for download at `www.mathworks.com/matlabcentral/fileexchange/14115-fa`) rather than the maximum likelihood-based algorithm *factoran*, which proved prohibitively inefficient on such a large dataset.

2.4.3 Evaluating model performance

Unless otherwise noted, model fitting was performed with 5-fold cross-validation (data are divided into 5 equally-size blocks, with 4 used for training and 1 for cross-

validation, with 5 non-overlapping cross-validation blocks). Because of the different natures of the simulated and experimental data, we use different measures to assess the quality of model fits on these different types of data sets.

To assess the quality of model fits on the simulated data we employed two different measures, one to evaluate the quality of the latent variables produced by the model and another to evaluate how well neural activity could be predicted. First, we measured the ability of each model to infer the true latent variables. For all models (RLVM, PCA, FA, ICA), model parameters were fit with the training data. Then, to calculate the latent variables on the cross-validation data, we used the activity of the neurons and the encoding matrices of each model (e.g., equation 2.14 for the RLVM) that were learned from the training data. Given that $\{\mathbf{z}^i\}_{i=1}^M$ are the M true latent variables and $\{\hat{\mathbf{z}}^j\}_{j=1}^P$ are the P latent variables inferred by a given model, the latent variable "maxcorr" measure is defined to be

$$\text{maxcorr} = \frac{1}{M}\sum_{i=1}^{M} \max_{j\in\{1,\ldots,P\}} |\text{corr}\left(\mathbf{z}^i, \hat{\mathbf{z}}^j\right)| \tag{2.19}$$

where corr is the Pearson correlation coefficient. The maxcorr simply measures the correlation between the inferred latent variable that best matches each true latent variable, averaged over the true latent variables. There is no restriction on the relationship between M and P; for example, if $M < P$, then the maxcorr measure will be close to one if each true latent variable is captured by at least one inferred latent variable.

The second measure of model performance for simulated data assessed how well the models could predict the overall population activity. For all models, model

parameters were fit with the training data for all neurons. Then, the activity predicted by the models on the cross-validation data was calculated using the encoding and decoding matrices of each model. The R^2 values reported are those obtained by comparing the true activity y_t^i of neuron i at time t with the activity predicted by the various methods $\hat{y}_t^i$ and averaging over all N neurons

$$R^2 = \frac{1}{N}\sum_{i=1}^{N}\left[1 - \frac{\sum_t (y_t^i - \hat{y}_t^i)^2}{\sum_t (y_t^i - \bar{y}^i)^2}\right] \tag{2.20}$$

where $\bar{y}^i$ is the average activity of neuron i.

For experimental data we do not have access to the "true" underlying latent variables, and thus cannot calculate a measure similar to the maxcorr measure presented above. We can, however, assess how well the models predict the overall population activity as in equation 2.20, with one caveat. It is possible with experimental data that a latent variable will capture the activity of a single neuron and thus inflate the resulting R^2 as calculated above. To address this issue, we employed a procedure similar to the leave-one-out prediction error introduced in [111]. For all models, model parameters were fit with the training data for all neurons. Then, to determine how well each model was able to capture the activity of a single neuron with the cross-validation data, we used the activit of all other neurons to calculate the activity of the latent variables (by setting the encoding weights of the left-out neuron to zero). We then performed a simple linear regression using the activity of the latent variables to predict the activity of the left-out neuron and used this prediction in the calculation of the R^2 measure in equation 2.20. Note that for this leave-one-out procedure if just a single neuron is contributing to the activity of a

latent variable, this procedure will result in a small R^2 value for that neuron during cross-validation.

2.4.4 Simulated data generation

We evaluated the performance of the RLVM using simulated data sets, which were generated with five non-negative latent variables that gave rise to the observed activity of 100 neurons. Note that these choices reflect our core hypotheses of the properties of latent variables in the cortex, and also match the assumptions underlying the RLVM model structure. Latent variables were generated by creating vectors of random Gaussian noise at 100-ms time resolution and then smoothing these signals with a Gaussian kernel. To enforce the non-negativity constraint on the latent variables, a positive threshold value was subtracted from each latent variable, and all resulting negative values were set to zero. Correlations between different latent variables were established by multiplying the original random vectors (before smoothing) by a mixing matrix that defined the correlation structure. Although smoothing and thresholding the correlated latent variables changed the correlation structure originally induced by the mixing matrix, the new correlation structures obtained by this procedure were qualitatively similar to those seen in experimental data.

The latent variables thus obtained acted as inputs to a population of neurons (figure 2.1). To calculate the coupling weights between the latent variables and the neurons in the population, a coupling matrix was created to qualitatively match the

coupling matrices found in experimental data (compare figures 2.1C and 2.4B). Since the experimental data used below in this article come from a two-photon imaging experiment, we chose to simulate data resembling two-photon fluorescence traces. To compute simulated fluorescence traces for each neuron, first the firing rate of the neuron was computed as the weighted sum of the latent variables, with the weights defined in the coupling matrix. The resulting firing rate was used to produce a spike train with a Poisson spike generator. The spike train was then convolved with a kernel to create the calcium signal, and finally Gaussian random noise was added to generate a simulated fluorescence signal. To ensure that our results were not dependent on a particular set of parameters used to generate the data, we also simulated data while varying the number of latent variables, the number of nonzero weights in the coupling matrix, and the signal-to-noise ratio (see figure 2.3).

2.4.5 Experimental data

Experimental protocol and data preprocessing. We evaluated the RLVM on data from the Svoboda lab [43], which have been made publicly available at `http://dx.doi.org/10.6080/K0TB14TN`. In this experiment mice performed a tactile discrimination task with a single whisker. During a given trial, the activity from neurons in layers 2/3 of barrel cortex expressing the GCaMP6s calcium indicator was recorded with two-photon imaging with three imaging planes set 15 μm apart. These imaging planes constituted a subvolume, and eight subvolumes were imaged during a given session. Furthermore, those same subvolumes were imaged

across multiple experimental sessions, and the resulting images were later registered so that activity of individual regions of interest (ROIs) could be tracked across the multiple sessions. Raw fluorescence traces were extracted from each ROI and neuropil corrected. For each ROI a baseline fluorescence F_0 was determined with a 3-minute sliding window and used to compute $\Delta F/F = (F - F_0)/F_0$.

Data selection. The publicly available data set contains the $\Delta F/F$ fluorescence traces of tens of thousands of neurons imaged over multiple sessions for eight different mice. To select subsets of this data for analysis with the latent variable models, we restricted our search to volume imaging experiments in which somatic activity was imaged in trained mice. We then looked for subsets of simultaneously imaged neurons that maximized the number of neurons times the number of trials imaged, selecting nine different sets of imaged neurons, three sets from each of three different mice. Within each set, neurons were removed from this selection if they met one or both of the following criteria: 1) $> 50\%$ of the fluorescence values were missing (indicated by NaNs); 2) the fluorescence trace had a signal-to-noise ratio (SNR) < 0.1. To estimate the SNR, a smoothed version of the fluorescence trace was estimated with a Savitzky-Golay filter (using MATLABs built-in function *sgolayfilt*). The noise was estimated using the residual between the original trace and the smoothed trace. The SNR was then calculated as the variance of the smoothed trace divided by the variance of the noise. After removal of individual neurons according to the above procedure, we then removed individual trials from the remaining data selection if NaN values existed in the whisker measurements or in one or more of the fluorescence traces. See table 2.1 for more information about

	Original Data			Analyzed Data	
Animal ID	Cell ID Range	ROI count	Trial count	ROI Count	Trial Count
an229716	21000-23464	1395	157	831	124
	33000-35351	1051	163	466	142
	18000-20377	1099	155	760	108
an229717	45000-47679	1695	173	406	146
	09000-11474	1364	156	805	106
	39000-41437	1313	136	466	089
an229719	**09000-11423**	**1358**	**162**	**685**	**098**
	15000-17478	1332	158	517	113
	18000-20489	1250	162	356	126

Table 2.1: **Experimental selection.** All experimental data used in figures 2.4-2.7 are from [43] and are publicly available at `http://dx.doi.org/10.6080/K0TB14TN`. Data analysis was performed on subsets of the data that contained a large number of neurons simultaneously imaged over many trials (see section 2.4.5). The Analyzed Data column shows the amount of data retained from the Original Data column after removal of neurons that had $> 50\%$ missing values in their fluorescence traces or had an estimated SNR < 0.1, as well as removal of trials that had missing values for any of the remaining fluorescence traces. The row in boldface corresponds to the data used for the analyses show in figures 2.4-2.6.

the specific subpopulations of neurons analyzed.

Alignment of fluorescence traces across sessions. As described above, the data from each experiment we used consisted of imaging the population activity over several recording sessions. Although fluorescence traces for each neuron were corrected for different baseline fluorescence levels in the online data set, we found it necessary to recalculate session-specific baseline fluorescence levels in order to concatenate traces across different sessions. [Unlike the analyses in the original work (Peron et al. 2015) [43], the models considered here were particularly sensitive to this baseline level because all fluorescence traces were analyzed jointly.] In the original

work, baseline fluorescence level was calculated using the skew of the distribution of raw fluorescence values, under the assumption that more active neurons will have more highly skewed distributions. However, this monotonic relationship breaks down for very active neurons, whose distributions are not as skewed since there are very few values near the baseline level. Because we found many neurons in the data set that fell into this last category, we recalculated baseline fluorescence levels on a session-by-session basis.

Using basic simulations of how the distribution of fluorescence values of a Poisson neuron depends upon its mean firing rate and SNR, we could match this with the data from each neuron to unambiguously infer its baseline fluorescence level. Specifically, for each neuron and each session, we measured the SNR of its fluorescence (described above) and also measured the skewness of its distribution of fluorescence (using MATLABs built-in function *skewness*). We simulated neural activity with the same SNR while varying the mean firing rate until the resulting distribution of values matched the measured skewness. Once the optimal mean firing rate was determined, we could then use the simulation to determine the best estimate of the baseline fluorescence level on a session-by-session basis. This procedure led to improved model estimation for all latent variable methods.

Sorting of coupling matrices. The ordering of simultaneously recorded neurons is arbitrary, so we chose the ordering for the display of the coupling matrices W to highlight groups of neurons that share similar coupling patterns. We first sorted the rows, using the coupling weights to the first latent variable (first column) for all neurons with a weight higher than a predefined threshold value.

We then sorted all remaining neurons with coupling weights to the second latent variable (second column) above the same threshold. This procedure is repeated for each column of W and produces a block diagonal structure (e.g. figure 2.4B). The last column is sorted without using the threshold so that it contains all remaining neurons.

Quantifying influence of individual latent variables on population activity. To determine the proportion of population activity driven by each latent variable in the experimental data sets, for each neuron we calculated the variance of the latent variable weighted by the neurons coupling strength to that latent variable, divided by the variance of the neurons measured activity, which was smoothed with a Savitzky-Golay filter to remove noise variance (implemented with the MATLAB function *sgolayfilt*). We considered a neuron to be driven by that latent variable if the proportion of total measured activity exceeded 0.10. To determine the proportion of predicted population activity, we performed a procedure similar to that above but divided by the variance of the predicted activity rather than the smoothed measured activity. [Note that, because latent variables can be correlated, these proportions will not add to 1 since we ignored cross-covariances.] These values were then averaged over all neurons to obtain a measure of the proportion of predicted activity driven by the given latent variable.

Chapter 3: The Generalized Affine Model (GAM)

3.1 Introduction

The activity of sensory neurons is highly variable in response to repeated presentations of the same stimulus [178, 179]. This response variability can potentially limit the amount of information contained in neural activity, since multiple stimuli could in principle elicit the same number of spikes [70]. Pooling the activity of multiple neurons can reduce the adverse effects of variability, but theoretical work has demonstrated that the efficacy of this strategy depends on how variability in neural responses is structured across the population [90, 92–95, 180–182]. These results have raised interest in uncoverng the structure of variability in experimentally-measured populations of neurons.

Much of the recent experimental work on understanding the structure of shared neural variability has focused on V1, where neural responses to simple stimuli are relatively well understood. Shared variability in V1 neural responses has been accounted for using a single additive factor, or latent variable, that modulates neural activity across the entire population [81, 82, 102]. Shared variability has also been modeled using a multiplicative latent variable that acts as a gain on the stimulus-driven response [67]. A more recent model, the "affine" model of Lin et al. [96]

(and the related multi-gain model of Arandia-Romero et al. [98]), combines both additive and multiplicative latent variables to account for neural variability, which describe more of the population response variability than either latent variable alone. Furthermore, Lin et al. demonstrated that the affine model was able to replicate essential features of the noise correlation structure in the population, including the dependence on stimulus tuning and stimulus orientation (a similar finding was reported in [102] using the average population activity as a single additive latent variable). In addition to these successes, the affine model is also appealing due to its analytical tractability. Both Lin et al. and Arandia-Romero et al. used their respective models to analytically explore the effect of the two latent variables on population coding.

Despite the affine model's ability to reconstruct essential features of the population response, and aid in the theoretical development of neural information coding, the model as implemented in these studies has some potential shortcomings. In Lin et al., the population coupling to the multiplicative latent variable is uniform across the population. In the multi-gain model of Arandia-Romero et al., population coupling to both latent variables is allowed to vary among the neurons, but the additive and multiplicative latent variables are constrained to be equal. It is currently unclear what effect these constraints have on the conclusions drawn by these studies. Furthermore, although both studies show that the affine model outperforms models with only an additive or multiplicative latent variable, it is not clear if a better model of response variability exists.

To test whether or not the affine model is a "good" model of V1 response

variability, we introduce a new modeling framework, the Generalized Affine Model (GAM), which allows each neuron to have a different coupling to distinct additive and multiplicative latent variables. We fit this model to simultaneously recorded neurons in anesthetized monkey V1 (the same data analyzed in Arandia-Romero et al.), as well as simultaneously recorded neurons in the LGN of awake and freely running mice, and find that this unconstrained affine model outperforms the constrained affine model of Lin et al. Using the V1 data, we show that this improved model is able to recapitulate findings made with the previous models, including the lack of correlation between stimulus tuning preferences and latent variable coupling [102], and a negative correlation in the modulatory effects of additive and multiplicative latent variables [98].

The GAM framework also allows us to simultaneously fit an arbitrary number of additive and multiplicative latent variables, which enables us to explore more complicated models of response variability. We find that by allowing several latent variables of each type, the GAM is able to outperform the unconstrained affine model (which has a single latent variable of each type). While this increase in performance is modest for the anesthetized V1 data, it is quite large for the awake LGN data. These findings suggest that the affine model is a good model of response variability in anesthetized V1, but must be extended in order to capture higher-dimensional variability outside of this context.

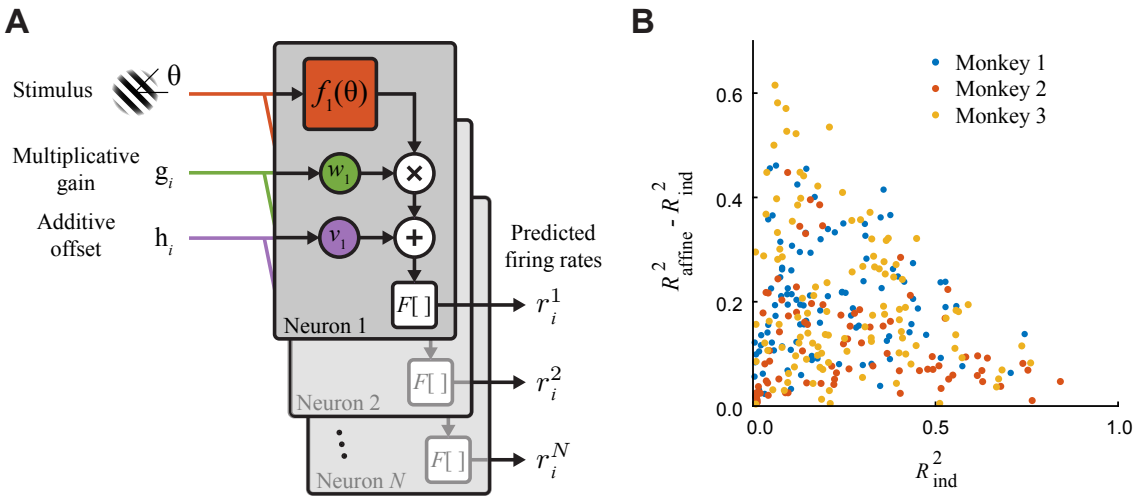

Figure 3.1: **The unconstrained affine model.** **A**: The response $f_n(\theta)$ of individual neurons to a drifting grating at angle θ is modulated by two latent variables on each trial i that are shared across the population: a multiplicative gain signal g_i and an additive offset h_i. $F[\cdot]$ is an optional spiking nonlinearity that can be applied to the output. **B**: The R^2 improvement of the affine model over the independent model, plotted as a function of the independent model R^2. The independent model only accounts for the effect of the stimulus on neural responses, and does not capture shared variability. Some neurons with poor stimulus models are well described by the affine model, indicating the large role that shared variability plays in driving V1 population responses.

3.2 Results

3.2.1 The unconstrained affine model outperforms constrained version

Trial-to-trial variability is a ubiquitous feature of neural responses, and a portion of this variability is shared across the population. We model this variability using both an additive latent variable h_i and a multiplicative latent variable g_i that are shared across the population and vary as a function of the trial i (figure 3.1A).

This model describes the firing rate r_i^n of neuron n on trial i as

$$r_i^n = (1 + w_n g_i) f_n(\theta) + v_n h_i \tag{3.1}$$

where $f_n(\theta)$ is the stimulus response of neuron n to a drifting grating in the direction of θ. This model generalizes several other models of V1 response variability. Goris et al. 2014 introduced a model that we will refer to as the "constrained multiplicative" model, where a single multiplicative latent variable uniformly modulates the stimulus response of the population:

$$\text{Constrained Multiplicative}: \quad r_i^n = (1 + g_i) f_n(\theta) \tag{3.2}$$

We will also define the "multiplicative" model to be an extension of this, where each neuron has its own weight to the latent variable:

$$\text{Multiplicative}: \quad r_i^n = (1 + w_n g_i) f_n(\theta) \tag{3.3}$$

and the analogous "additive" model:

$$\text{Additive}: \quad r_i^n = f_n(\theta) + v_n h_i \tag{3.4}$$

Lin et al. 2015 introduced a model that we will refer to as the "constrained affine" model, which contains both an additive and multiplicative latent variable. Each neuron has its own weight to the additive latent variable, but all neurons are constrained to be uniformly modulated by the multiplicative latent variable:

$$\text{Constrained Affine}: \quad r_i^n = g_i f_n(\theta) + v_n h_i \tag{3.5}$$

A variant of this model, the "multi-gain" model, was introduced in Arandia-Romero et al. 2016, which allows each neuron to have its own weights to each latent variable,

but constrains the additive and multiplicative latent variables to be equal:

$$\text{Multi-gain}: \quad r_i^n = (1 + w_n g_i) f_n(\theta) + v_n g_i \tag{3.6}$$

We refer to our formulation in equation 3.1 as the "affine" model, which does not constrain either weights or latent variables to be equal. Following Lin et al. 2015, we define the "independent" model to be:

$$\text{Independent}: \quad r_i^n = f_n(\theta) \tag{3.7}$$

which does not model shared variability, and serves as a baseline model to gauge the impact of shared variability on population responses.

We first demonstrate that the affine model is able to capture a significant fraction of the trial-to-trial variability in the activity of V1 neurons in anesthetized macaques in response to drifting gratings (figure 3.1B; see section 3.4.1 for experimental details). The improvement in model R^2 between the affine model and the independent model can be quite large across all three monkeys, and even neurons with poor stimulus models can be well described by the affine model. This suggests that these neurons are not in fact as noisy as their stimulus models would indicate; rather, they are predominantly driven by inputs that are not locked to the stimulus, and demonstrate that activity in primary visual cortex does not simply encode the external stimulus. For the remainder of this chapter we use the Quality Index of Lin et al. to quantify model performance, which is defined as

$$\text{QI}_{\text{model}} = \frac{R^2_{\text{model}} - R^2_{\text{ind}}}{1 - R^2_{\text{ind}}} \tag{3.8}$$

a rescaled version of the R^2 measure such that $QI = 0$ for a model with predictions

that are equivalent to the independent model, and $QI = 1$ for perfect prediction.

We next asked how the different constraints of previous models affect model performance. In particular, we were interested if allowing each neuron to have their own coupling to the multiplicative latent variable improved the performance of the models. We found this to be true when comparing the multiplicative and constrained multiplicative models (figure 3.2A; p<5e-5 for each of the three monkeys, two-sided sign test) as well as when comparing the affine and constrained affine models (figure 3.2B; p<5e-5 for each of the three monkeys). Furthermore, the affine model outperformed both the additive (figure 3.2C; p<5e-10 for each of the three monkeys) and multiplicative models (figure 3.2D, p<5e-10 for each of the three monkeys).

The unconstrained affine model is indeed a better model of V1 response variability than previous models, due to a more general model structure. Because interpretation of model parameters in general depends on model structure, we next asked whether or not the affine model could reproduce some of the previous results obtained with constrained models.

3.2.2 Affine model recapitulates results of previous models

We first looked to verify a result from Okun et al. 2015 [102], which showed that variability in the responses of V1 neurons could be described by their coupling to the average population activity, and that the degree of population coupling was independent of tuning properties. This intriguing finding suggests that neural

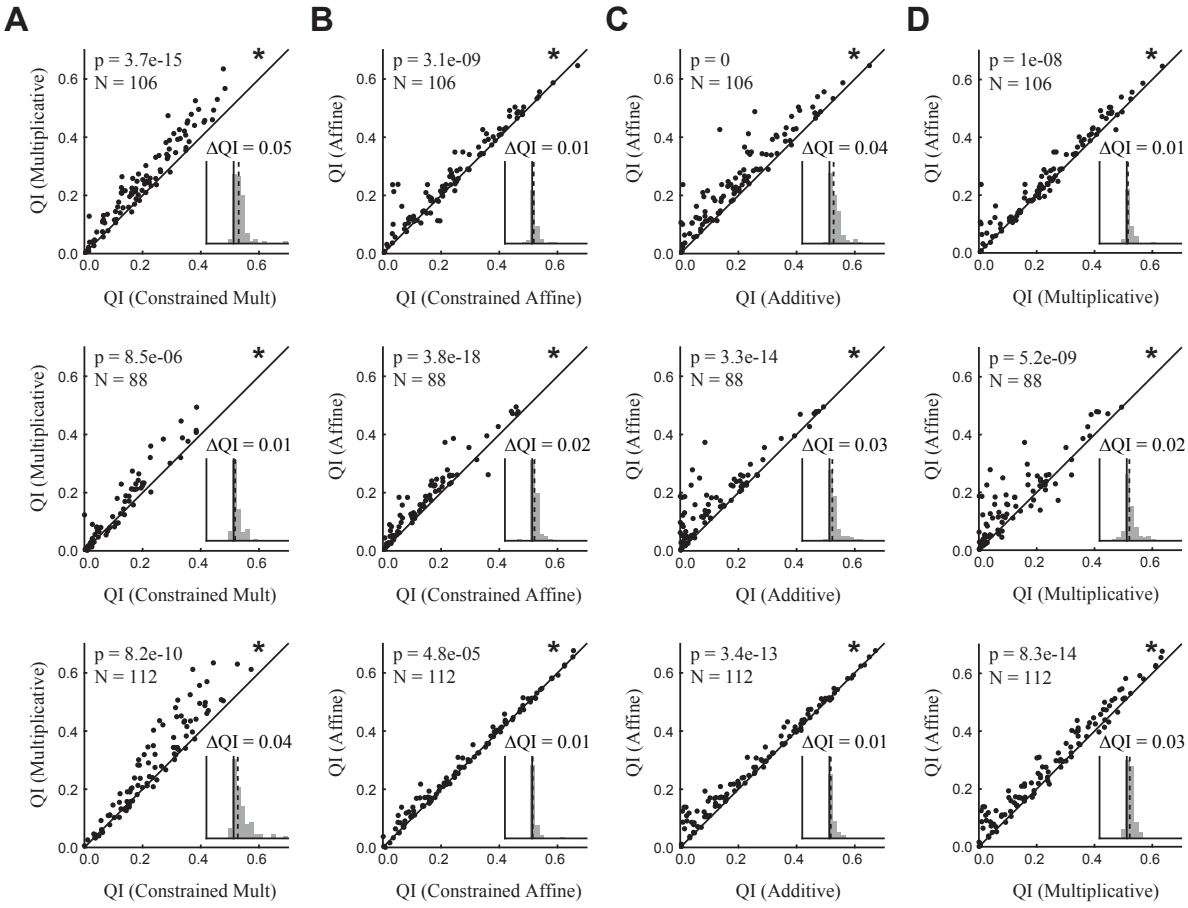

Figure 3.2: **Unconstrained affine model performance on V1 data.** A comparison of the cross-validated performance of the models on V1 data across three anesthetized monkeys (*top*, *middle* and *bottom* for each panel). Performance is measured by the Quality Index, which is 0 if the model is equivalent to the independent model, and 1 for perfect prediction. Each point is the mean Quality Index over 10 cross-validation folds. **A**: The multiplicative model outperforms the constrained multiplicative model, which requires all neurons to have the same weight to the multiplicative latent variable. **B**: The affine model outperforms the constrained affine model, which also requires all neurons to have the same weight to the multiplicative latent variable, but allows different weights to the additive latent variable. **C**: The affine model outperforms the unconstrained additive model. **D**: The affine model outperforms the unconstrained multiplicative model. p-values are calculated using the two-sided sign test, and the asterisk location relative to the diagonal indicates the model with the significantly higher median value.

		Ori Pref/Mult		Ori Pref/Add			Dir Pref/Mult		Dir Pref/Mult	
	N_{ori}	ρ	p-value	ρ	p-value	N_{dir}	ρ	p-value	ρ	p-value
Monkey 1	100	0.17	0.10	-0.02	0.88	81	-0.09	0.40	-0.03	0.77
Monkey 2	85	-0.08	0.46	-0.18	0.11	71	0.04	0.77	0.01	0.90
Monkey 3	109	0.14	0.14	-0.13	0.16	88	0.05	0.67	0.34	1.1e-3
Combined	294	0.02	0.71	-0.11	0.05	240	0.11	0.08	0.13	0.04

Table 3.1: **Correlation between affine model coupling weights and tuning properties.** Correlation coefficients and p-values for individual datasets; the combined datasets are shown in figure 3.3. Only neurons with orientation and direction selectivty indices >0.05 are used in this analysis.

activity, even in early visual areas, can be modulated in a stimulus-independent manner by fluctuations in cortical network activity. However, our affine model far outperforms a similar additive model (figure 3.2C) due to the different form of the additive latent variable (a weighted average of the population activity rather than the average; see section 3.4.2) and the incorporation a multiplicative latent variable.

We were interested if this result still holds given a more accurate model of the response variability. We found that there was a wide range of coupling weights to both the multiplicative and additive latent variables, and that these weights are in fact largely uncorrelated with either orientation preference or direction preference (combined results in figure 3.3; individual results in table 3.1).

We next investigated two results from Arandia-Romera et al. concerning the differential effects of the additive and multiplicative latent variables on response variability. They found, using a model-free analysis, that the additive and multiplicative factors modulating neural responses were negatively correlated. We find a similar effect using the unconstrained affine model, where the two sets of coupling

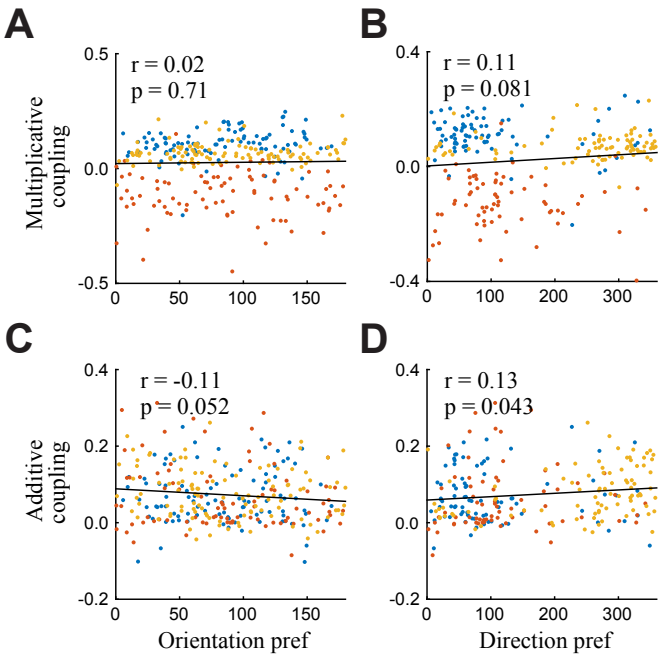

Figure 3.3: **Coupling to latent variables is uncorrelated with tuning properties.** The orientation preference of individual neurons is not significantly correlated with either the coupling weights to the multiplicative latent variable (**A**) or the additive latent variable (**C**). Only neurons with an orientation selectivity index >0.05 are used in this analysis. The direction preference of individual neurons is not significantly correlated with the multiplicative coupling weights (**B**), and just barely significantly correlated with the additive coupling weights (**D**). Only neurons with a direction selectivity index >0.05 are used in this analysis. Colors indicate different datasets, and table 3.1 displays correlation coefficients and p-values for individual datasets.

weights are negatively correlated (figure 3.4A), which is signficant in two of the three monkeys (p<5e-7).

The multi-gain model of Arandia-Romero et al. constrains the additive and multiplicative latent variables to be equal (equation 3.6). Our unconstrained affine model does not impose this constraint, and we were interested in whether or not this feature of the multi-gain model is replicated in our model fits. We found that there is in general a strong correlation between the additive and multiplicative latent variables, but still a large amount of difference between the two (figure 3.4B). It is

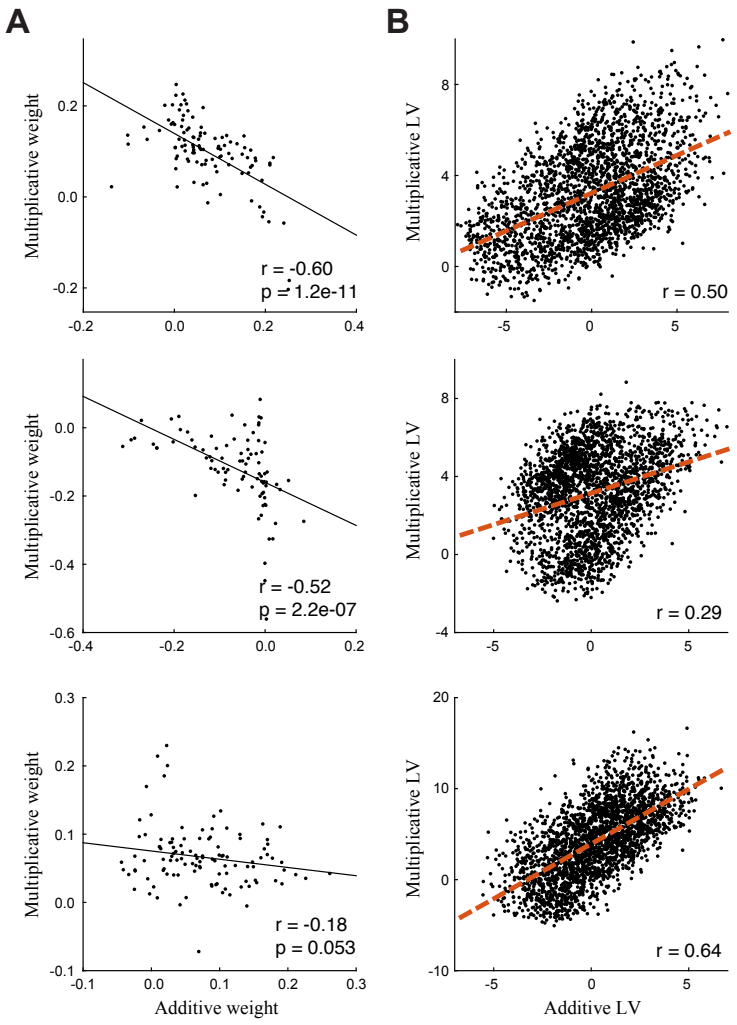

Figure 3.4: **Effects of additive and multiplicative latent variables are negatively correlated. A**: Neuron coupling weights in the affine model are negatively correlated, which recapitulates a result from the model-free analysis of Arandia-Romero et al.; points represent average weight over 10 cross-validation folds. **B**: Values of the inferred latent variables are positively correlated. Values are calculated using the cross-validation data.

not clear whether constraining these latent variables to be the same would increase model performance (in which case the differences in values are due to overfitting) or decrease it (in which case these latent variables represent two distinct, albeit correlated signals), which is a direction for future work.

3.2.3 LGN activity in awake mice is additive and multiplicative

To further test if the affine model is appropriate for describing response variability in the early stages visual processing, we used two-photon imaging to record activity from LGN boutons in primary visual cortex of awake and freely running mice in response to drifting gratings (see section 3.4.1 for experimental details).

Like the anesthetized macaque V1 data, we found that allowing each neuron to have its own coupling to the multiplicative latent variable significantly improved model fits in both the multiplicative models (figure 3.5A; p<5e-4 for three of the four mice; p>0.05 for the remaining; two-sided sign test) and the affine models (figure 3.5B; p<5e-3 for each of the four mice). Also as before, the affine model outperformed both the additive (figure 3.5C; p<5e-2 for each of the four mice) and multiplicative models (figure 3.5D, p<5e-5 for each of the four mice).

Previous work has demonstrated that a large portion of variability in anesthetized macaque V1 is driven by low-frequency fluctuations induced by the anesthesia [81]. In our experiment, mice were awake and freely running, which allowed us to investigate additional drivers of variability. For example, locomotion can modulate responses in rodent primary visual cortex [154, 183, 184] and LGN [185, 186]. Pupil size, which can serve as a proxy for arousal state [187], is also correlated with variability in primary visual cortex of awake mice [188]. Similarly, we have previously reported a strong correlation between population activity and pupil size in this data from LGN [189].

To understand whether or not these factors can describe a similar amount of

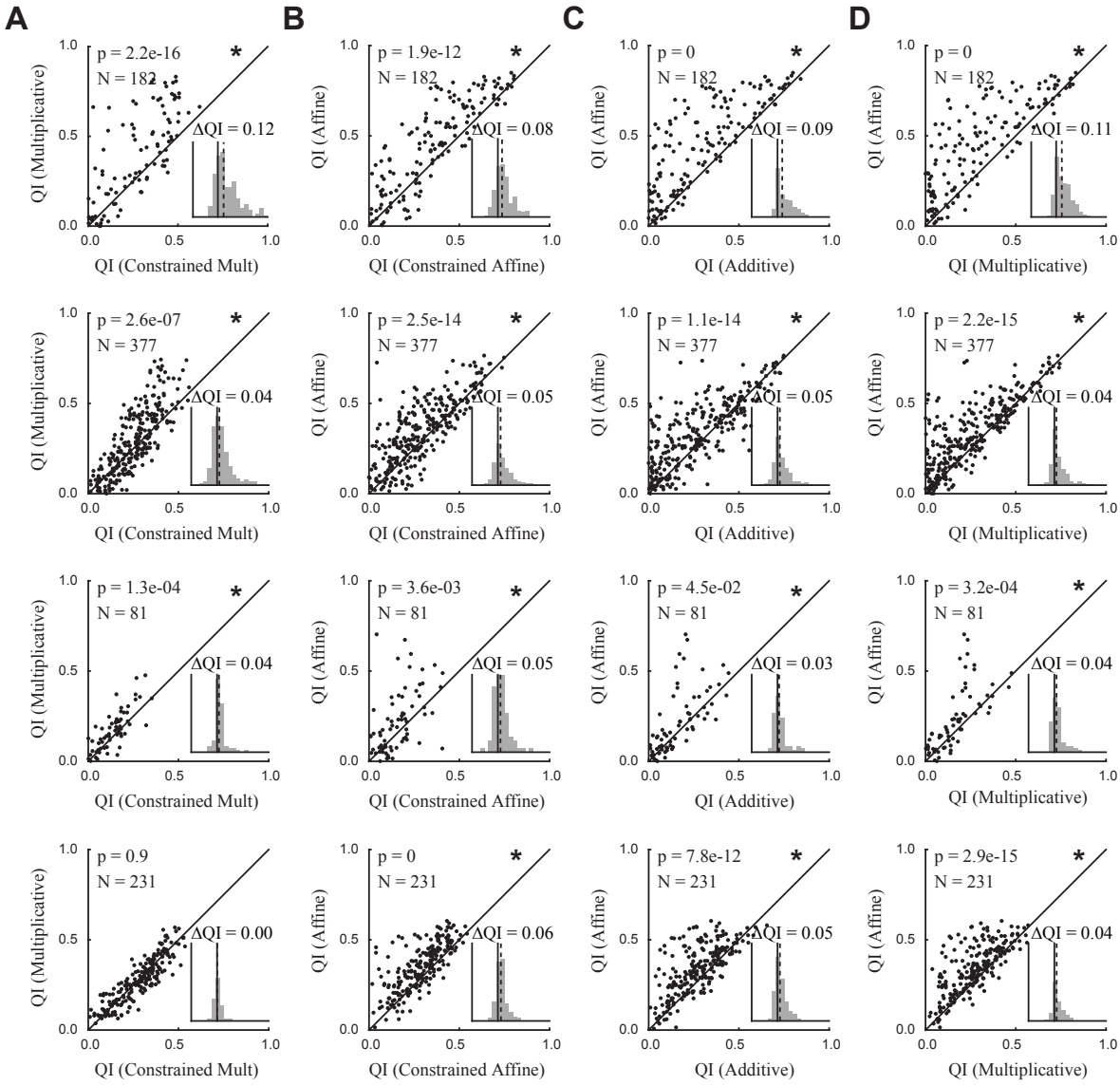

Figure 3.5: **Unconstrained affine model performance on LGN data.** Comparison of the cross-validated performance of the models on LGN data across four awake and freely running mice, which demonstrate the significant performance increases that come from allowing each neuron to have its own coupling to the multiplicative latent variable in both the multiplicative (**A**) and affine (**B**) models. Additionally, the affine model performs signficantly better than using either additive (**C**) or multiplicative (**D**) latent variables alone. Same conventions as figure 3.2.

variability as the latent variables that we infer with our model, we fit the unconstrained additive, multiplicative and affine models using four sets of predictors: 1) the latent variable inferred from the high-dimensional population activity (which is a weighted average of the population activity, and the predictor used in the models in figure 3.5); 2) the average population activity; 3) the pupil diameter of the animal; and 4) the running speed of the animal. We found that the inferred latent variable performed significantly better than the other three predictors in almost every comparison (additive: $p<0.05$ in 11/12 models; multiplicative: $p<0.05$ in 11/12 models; affine: $p<0.005$ in 12/12 models; two-sided sign test). These results demonstrate that although population activity in LGN is correlated with these factors, they are not as predictive of the trial-to-trial variability in our framework. This reflects the fact that an arousal signal, for example, might modulate both pupil diameter and LGN activity (hence leading to their correlation), but their resulting relationship does not necessarily need to be linear.

3.2.4 Is the affine model a good description of early visual responses?

We have demonstrated that the unconstrained affine model captures more response variability than related models in both anesthetized monkey V1 (figure 3.2) and awake mouse LGN (figure 3.5). However, this does not preclude the possibility that there are better models of the neural population response that can capture more of the variability. To address this question we introduce two new classes of models and compare their performance to the unconstrained affine model on both

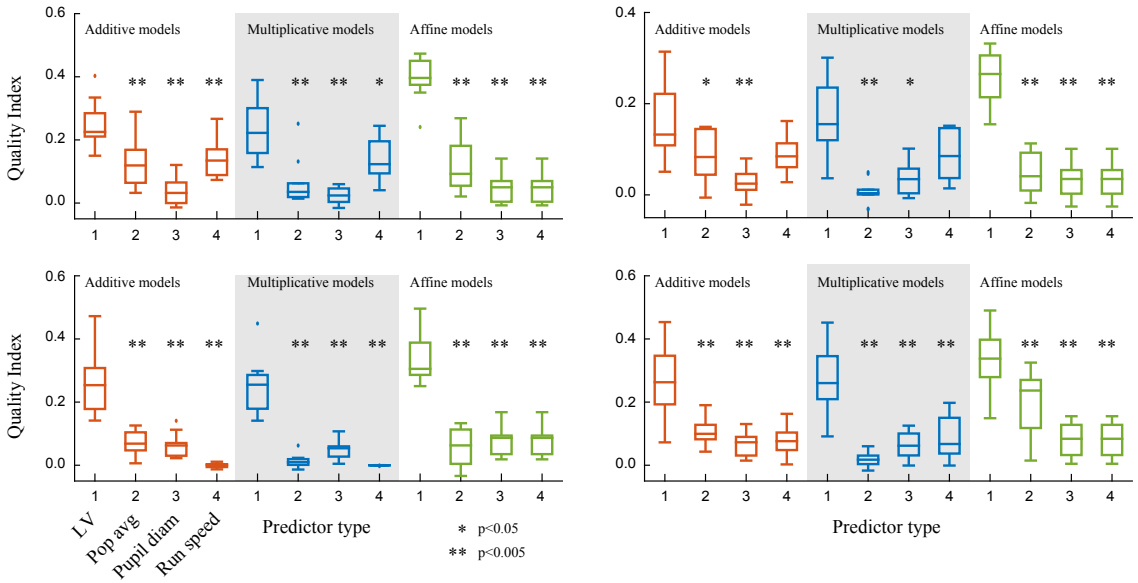

Figure 3.6: **Latent variables predict LGN responses better than experimental observables.** Comparison of the cross-validated performance of the models on LGN data when using different predictors for the multiplicative gain and/or additive offset in the additive model (*red boxplots*), multiplicative model (*blue boxplots*), and affine model (*green boxplots*) for four mice. The predictors are the 1) latent variable inferred from the high-dimensional population activity; 2) average population activity; 3) pupil diameter; and 4) running speed. Asterisks indicate if model performance is significantly less than that using the latent variable predictor, within a particular model class (e.g. additive). Comparisons between additive and multiplicative models, for example, are omitted. $^*p<0.05$, $^{**}p<0.005$, two-sided sign test.

datasets.

The first model extends the unconstrained affine model by allowing an arbitrary number of additive and multiplicative latent variables, which we call the generalized affine model (GAM):

$$r_i^n = \left(1 + \sum_{k=1}^{K} w_n^k g_i^k\right) f_n(\theta) + \sum_{m=1}^{M} v_n^m h_i^m \tag{3.9}$$

Each neuron is allowed to have its own coupling to each of the M additive latent variables, and a different coupling to each of the K multiplicative latent variables (see section 3.4.2 for model fitting details); the unconstrained affine model is recovered

from this model when $M = K = 1$.

In the V1 dataset we found that the best cross-validated model included multiple additive and multiplicative latent variables for each of the three monkeys (figure 3.7A). However, the number of additional latent variables in the optimal models was small (1-3 for both types), and the Quality Index, while significantly greater than that of the affine model for two of the three monkeys, was not large in magnitude. This result indicates that the affine model, with a single additive and multiplicative latent variable, is indeed a good description of the population response in anesthetized V1.

To further test the affine model, we compared it to a general nonlinear latent variable model that is agnostic to the stimulus identity on a given trial. We used an autoencoder neural network (see section 3.4.3), which models the neural responses as

$$r_i^n = f_n^{\text{auto}}(\mathbf{z}_i) \tag{3.10}$$

where f_n^{auto} is a single- or multi-layer neural network (RLVM and SRLVM, respectively) and $\mathbf{z}_i \in \mathbb{R}^K$ is a vector of Q latent variables. Unlike the previous models, where latent variables interacted with the stimulus response in a specific way, this model does not explicitly model the stimulus response and makes no assumptions on what the $\mathbf{z}_i$ represent or how they are combined (and as such, they will represent elements of both the stimulus response and the trial-to-trial variability).

We found that the SRLVM performed as well as or worse than the unconstrained affine model and the optimal GAMs for each monkey, across a range of

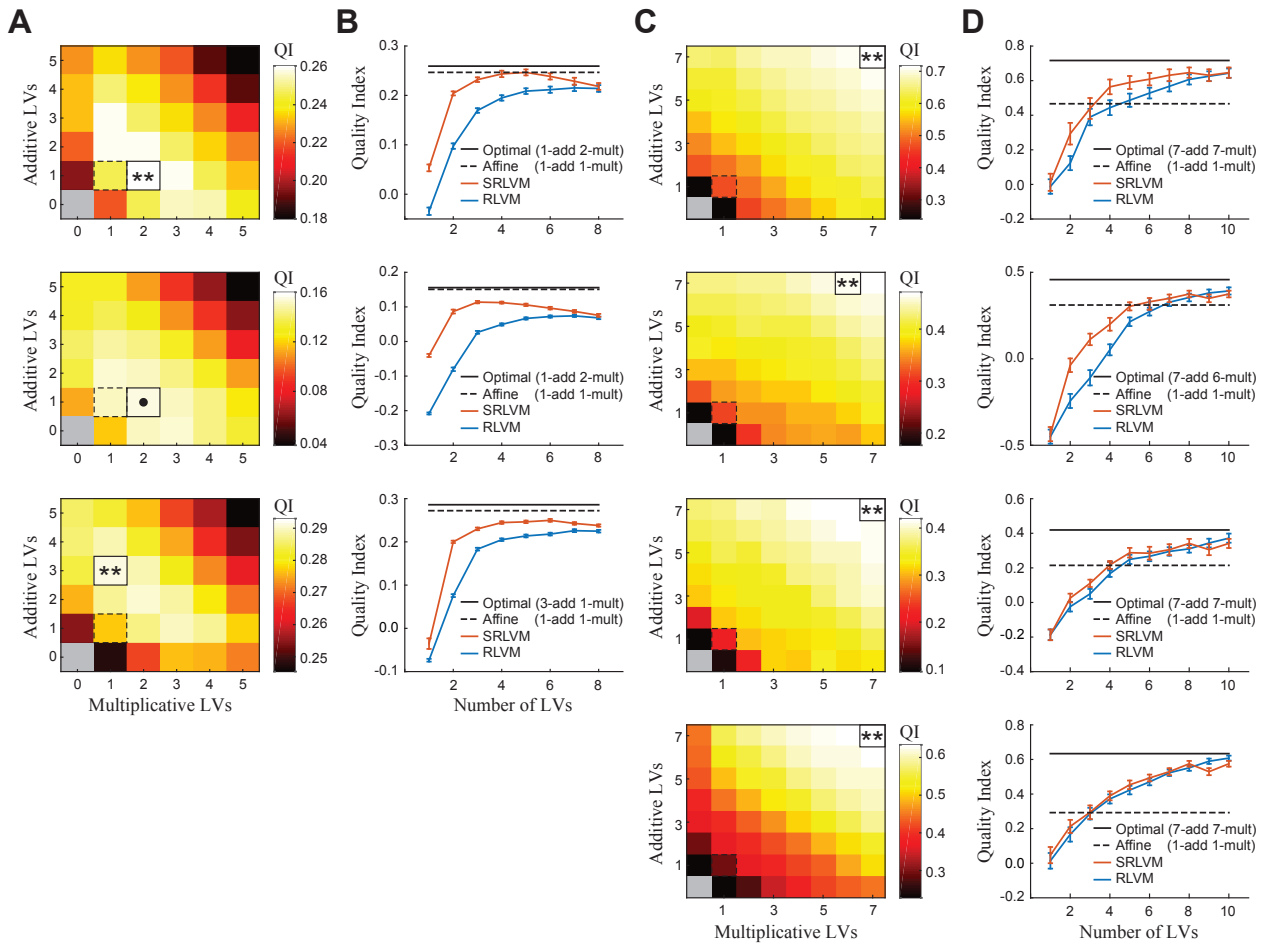

Figure 3.7: **Generalized affine model performance on V1 and LGN data.** Model performance using V1 data from anesthetized monkey (panels *A, B*) and LGN data from awake mouse (panels *C, D*). **A, C**: Mean QI over neurons and cross-validation folds for the generalized affine model, which allows an arbitrary number of additive and multiplicative latent variables. The standard affine model is outlined (*dashed black line*) as well as the generalized affine model with the highest QI (*solid black line*; $^{**}p<0.005$, $\bullet p>0.05$, two-sided sign test on the mean compared to the affine model). Even with seven additive and seven multiplicative latent variables, the performance of the generalized affine model for the LGN data continued to increase (*C*). **B, D**: Performance of the affine model and optimal generalized affine model, plotted against the RLVM and SRLVM, which describe the data using a linear (RLVM) or arbitrary nonlinear (SRLVM) combination of latent variables, and do not include a stimulus model.

latent variables (figure 3.7B). Though we did not perform an exhaustive search over the hyperparameters governing the autoencoder neural networks, this result further supports our earlier claim that the affine model is indeed a good description of shared variability in this V1 data.

We next analyzed the LGN data in the same manner, and found strikingly different results. The GAM far outperformed the standard affine model in all four mice (figure 3.7C). Furthermore, the optimal GAM utilized six or seven of each latent variable type, the maximum number that we tested. Consistent with this result, the RLVM and SRLVMs also outperformed the affine model, and only started to saturate in performance with around ten latent variables (figure 3.7D). The relatively large number of latent variables needed to describe this variability indicates that the standard affine model is not a good description of variability in this LGN data.

3.3 Discussion

We extended the affine models of neural population response variability developed in Lin et al. 2015 and Arandia-Romero et al. 2016 to allow each neuron to have its own coupling to separate additive and multiplicative latent variables. We showed that this unconstrained affine model is better at capturing variability than previous models in anesthetized macaque V1 (figure 3.2) and awake and freely running mouse LGN (figure 3.5). We were also able to replicate several previous findings using this improved model, demonstrating that coupling to additive and multiplicative terms was uncorrelated with tuning preferences (figure 3.3) and that

the effects of additive and multiplicative latent variables were negatively correlated (figure 3.4).

We also introduced two new models of population activity in order to better gauge the performance of the unconstrained affine model. The first model, the generalized affine model, allows for an arbitrary number of additive and multiplicative latent variables, and our model fitting framework allows us to fit all parameters and infer all latent variables simultaneously. The second model, the Stacked Rectified Latent Variable Model (SRLVM), is a standard multi-layer autoencoder neural network that we used as a general nonlinear latent variable model. We found that these models were only slightly better at explaining response variability than the affine model in anesthetized macaque V1 (figure 3.7), but far outperformed the affine model in awake mouse LGN.

Anesthesia likely plays an important role in the discrepancy between our results in V1 and LGN. Neural activity under anesthesia can be quite different than in the awake and behaving state. [81] found a single additive latent variable could explain a large portion of the noise correlations induced by anesthesia in monkey V1, which is supported by our analysis (figure 3.7B). The awake and behaving state appears to contain a richer variety of population response patterns, which requires more latent variables to accurately describe. However, these datasets also differed in visual area, species, and recording modality. Uncovering the role of anesthesia in these results will require analyzing data from awake monkey V1.

Lin et al. applied their affine model to activity from the primary visual cortex of quietly awake mice, and found that it was able to explain more response variability

than both the additive and multiplicative models. This finding is consistent with our results in figure 3.5. Additionally, we showed that the latent variables inferred by the affine model were able to explain significantly more variability than experimental observables like pupil diameter and running speed (figure 3.6).

However, by introducing a more complicated model structure, we were able to demonstrate that a large portion of the affine model's residual variability is in fact explainable as well. Determining how this higher-dimensional variability influences sensory processing will be more difficult than analyzing a single additive or multiplicative latent variable. Moreover, the experiments analyzed here do not involve a behavioral component (besides locomotion in the LGN data), which limits the extent to which we can address the functional role of these various latent variables [190].

Using these models of population response variability in settings with behavioral components will shed light on how these different forms of shared variability might influence behavior. For example, [135] were able to derive a latent variable that was predictive of animal performance on a trial-to-trial basis during an attention task, establishing an important link between neural variability and behavior. To analyze that same data, Rabinowitz et al. 2015 [101] developed a model that contains three different classes of multiplicative gain terms, similar to equation 3.9, and found that a small number of gain terms could describe the response variability of the full population. Additionally, they were able to show that these gain terms were predictive of behavioral performance, and dependent on the reward outcome of the previous trial. By reducing high-dimensional population activity to a small

number of latent variables, the models introduced here (as well as those this work builds upon) will become increasingly useful for understanding how the activity from large groups of neurons contextually process external stimuli in behaviorally relevant settings.

3.4 Methods

3.4.1 Experimental data

V1 dataset. We analyzed electrophysiology data from the Kohn lab, which has been made publicly available at `http://dx.doi.org/10.6080/K0NC5Z4X`. Spiking activity was recorded with a Utah array in primary visual cortex from three anesthetized macaques, in response to full-contrast drifting gratings with 12 equally-spaced orientations, presented for 1280 ms (200 repeats). Full details can be found in [191]. Spike counts were analyzed using a single 500 ms time bin per trial (500 ms to 1000 ms after stimulus onset). Spike counts were square-rooted before fitting the models to stabilize variance and reduce the influence of neurons with high firing rates [111].

LGN dataset. We analyzed two-photon imaging data from simultaneously recorded presynaptic, putative thalamic (LGN) boutons in mouse primary visual cortex. The mice were awake and head-fixed, and allowed to freely run on a treadmill as full-field drifting gratings of 12 different directions and orientations were displayed for five seconds, interspersed with five seconds of a blank gray screen (10 repeats). Animal velocity and pupil diameter were also recorded throughout the

session, and resampled at the frame rate of the two-photon microscope for further analysis. Full details can be found in [189]. The data were analyzed at the resolution of whole trials, so that fluorescence values were separately averaged over each stimulus presentation period and each blank period.

Calculating tuning properties. The preference of each unit to the direction of the drifting gratings was quantified by first calculating the average response $\bar{y}_\theta$ to grating direction θ (in radians), and the direction preference (DP) is then defined as

$$\text{DP} = \text{Arg}\left(\frac{\sum_\theta \bar{y}_\theta \exp(i\theta)}{\sum_\theta \bar{y}_\theta}\right) \tag{3.11}$$

The direction selectivity index (DSI) quantifies the strength of this preference as

$$\text{DSI} = \text{Abs}\left(\frac{\sum_\theta \bar{y}_\theta \exp(i\theta)}{\sum_\theta \bar{y}_\theta}\right) \tag{3.12}$$

The preference of each unit to the orientation of the drifting gratings was also quantified. The orientation of the grating will be the same for two gratings drifting in opposite directions, and so the measure is defined similarly:

$$\text{OP} = \frac{1}{2}\text{Arg}\left(\frac{\sum_\theta \bar{y}_\theta \exp(i2\theta)}{\sum_\theta \bar{y}_\theta}\right) \tag{3.13}$$

as well as the orientation selectivity index:

$$\text{OSI} = \text{Abs}\left(\frac{\sum_\theta \bar{y}_\theta \exp(i2\theta)}{\sum_\theta \bar{y}_\theta}\right) \tag{3.14}$$

3.4.2 The Generalized Affine Model

The GAM models neural responses as the result of a gain term that is shared across the population which multiplicatively modulates stimulus processing, along

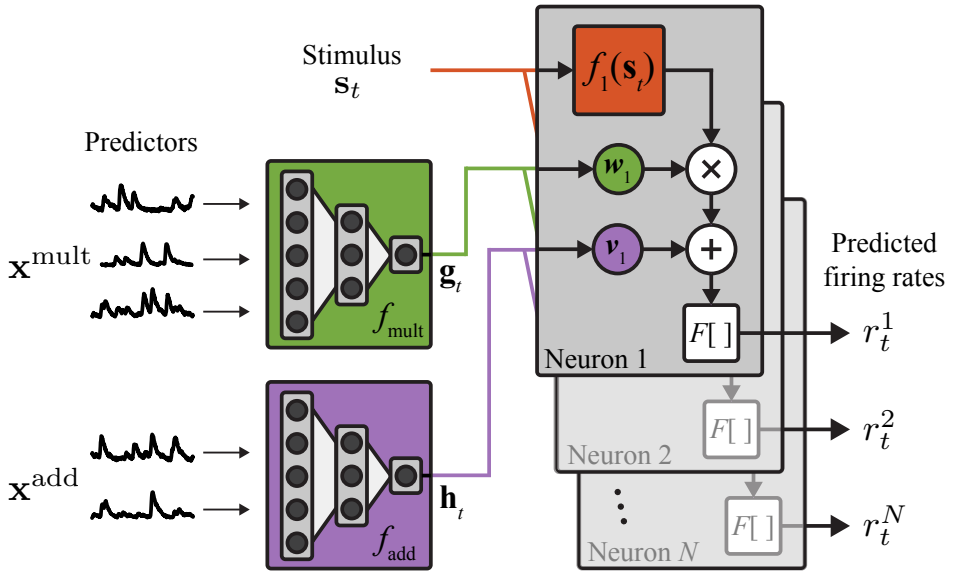

Figure 3.8: **Structure of the Generalized Affine Model.** A structured, nonlinear latent variable model that uses a neural network to transform a set of predictors (such as population activity) into a multiplicative gain signal (*green network*). This signal is shared across the entire population and modulates the output of a stimulus model (*red box*) for each individual neuron. A separate neural network transforms its input into an additive signal (*purple network*) that is also shared across the population.

with an additive term that is likewise shared across the population (figure 3.8). We define the full model for the predicted firing rate of each neuron n at time t, r_t^n, in several steps.

We first define a stimulus model $f_n(\mathbf{s}_t)$ for each neuron n such that $f_n(\cdot)$ maps $\mathbf{s}_t$, the vector of stimulus values at time t (and perhaps previous time lags), to a rate at each time t. f_n could be as simple as a one-dimensional tuning curve or a peri-stimulus time histogram (PSTH), or a much more complicated stimulus processing model like the hierarchical models described in section 1.4.3. We denote the parameters of the stimulus models $\{f_n\}_{n=1}^N$ as θ_{stim}. For this basic stimulus

model, the GAM estimates the firing rate r_t^n as

$$r_t^n = F[c_n + f_n(\mathbf{s}_t)] \tag{3.15}$$

where c_n is an overall offset term and $F[\cdot]$ is a pointwise spiking nonlinearity.

Next we define latent variables $\mathbf{g}_t \in \mathbb{R}^K$ that have an explicitly multiplicative effect on the stimulus processing (figure 3.8 only illustrates a single multiplicative latent variable, but in principle more than one can be fit using this framework). These latent variables are shared across the entire population, though each neuron has its own weight w_n^k to each of the K latent variables g_t^k:

$$r_t^n = F\left[c_n + u\left(\sum_{k=1}^{K} w_n^k g_t^k + b_n\right) f_n(\mathbf{s}_t)\right] \tag{3.16}$$

where b_n is a bias term and $u(\cdot)$ is a static nonlinearity. We use $u(x) = 1 + x$ to fit the models in this chapter, though other functions like $u(x) = \exp(x)$ are also suitable.

Inference of the latent variables is performed by using a neural network f_{mult} to nonlinearly map a set of predictors $\mathbf{x}_t^{\text{mult}}$ into the latent variables:

$$\mathbf{g}_t = f_{\text{mult}}(\mathbf{x}_t^{\text{mult}}) \tag{3.17}$$

We denote the parameters for the multiplicative latent variables (w_n^k, b_n, and the weights and biases of f_{mult}) as θ_{mult}.

Finally, we allow the GAM to have additive latent variables $\mathbf{h}_t \in \mathbb{R}^M$ as well, in order to capture activity that cannot be accounted for by the modulated stimulus model of equation 3.17. Like the multiplicative latent variables, these additive latent variables are shared across the population, but each neuron has its own weight v_n^m

to each of the M latent variables h_t^m

$$r_t^n = F\left[c_n + u\left(\sum_{k=1}^{K} w_n^k g_t^k + b_n\right) f_n(\mathbf{s}_t) + \sum_{m=1}^{M} v_n^m h_t^m\right] \tag{3.18}$$

Inference of the additive latent variables likewise uses a neural network,

$$\mathbf{h}_t = f_{\text{add}}(\mathbf{x}_t^{\text{add}}) \tag{3.19}$$

and we denote the parameters for the additive latent variables (v_n^m and the weights and biases of f_{add}) as θ_{add}.

To fit the parameters $\{\theta_{\text{stim}}, \theta_{\text{mult}}, \theta_{\text{add}}\}$ of the GAM we define the loss function to be the penalized negative log-likelihood $\mathcal{L}_{\text{GAM}}$ under the chosen observation model (Gaussian or Poisson). In the models fit in this chapter we use the Gaussian observation model (with identity covariance), so that the cost function becomes

$$\mathcal{L}_{\text{GAM}} = \frac{1}{2T}\sum_{t=1}^{T} \left\|\mathbf{y}_t - \mathbf{r}_t\right\|_2^2 + \lambda_{\text{stim}} q(\theta_{\text{stim}}) + \lambda_{\text{add}} q(\theta_{\text{add}}) + \lambda_{\text{mult}} q(\theta_{\text{mult}}) \tag{3.20}$$

where the $q(\cdot)$ are regularization terms that we take to be the L_2 norm on the weights, governed by the hyperparameter λ.

The parameters θ_{mult} and θ_{add} are initialized using the varimax-rotated principal components (see section 3.4.3); θ_{stim} is initialized by fitting the desired stimulus processing model f_n (equation 3.15) to each neuron individually and using the resulting parameters. To train the full model we hold θ_{stim} fixed and simultaneously optimize θ_{mult} and θ_{add} using the L-BFGS optimization routine [177]. Models were fit to training data using a range of λ values (we constrained $\lambda_{\text{add}} = \lambda_{\text{mult}}$), and the value that resulted in the smallest cost on the held-out cross-validation fold was chosen.

For the V1 data, we modeled the stimulus response of each neuron using a tuning curve, for a total of 12 parameters. f_{mult} had no hidden layers, and the output unit implemented $h(x) = 1 + x$ (no bias was fit). f_{add} likewise had no hidden layers, and used a linear unit for the output.

For the LGN data, we modeled the stimulus response of each neuron as a tuning curve, with one learnable parameter for each stimulus identity and another learnable parameter for each associated blank period, for a total of (12 stimuli + 1 blank control) $\times$ 2 (stimulus/blank periods) = 26 parameters. We fit models using a range of predictors for $\mathbf{x}_t^{\text{mult}}$ and $\mathbf{x}_t^{\text{mult}}$, including the average population response, pupil diameter, running speed, and the full-dimensional population response (figure 3.6). For all predictors, f_{mult} and f_{add} were the same as those described for the V1 data.

3.4.3 The Stacked Rectified Latent Variable Model (SRLVM)

The Stacked Rectified Latent Variable Model (SRLVM) extends the RLVM introduced in chapter 2 to multiple hidden layers. The SRLVM constrains the latent variables $\mathbf{z}_t$ to be some encoding function f_{enc} of the population activity $\mathbf{y}_t$; the population activity can then be coupled to the latent variables with a (in general different) decoding function f_{dec} [142, 168]. We consider the cases where both f_{enc} and f_{dec} are implemented with a single affine transformation (RLVM), and where both f_{enc} and f_{dec} are implemented with feedforward neural networks (SRLVM). Under the autoencoder model the inferred latent variables $\mathbf{z}_t$ and predicted population

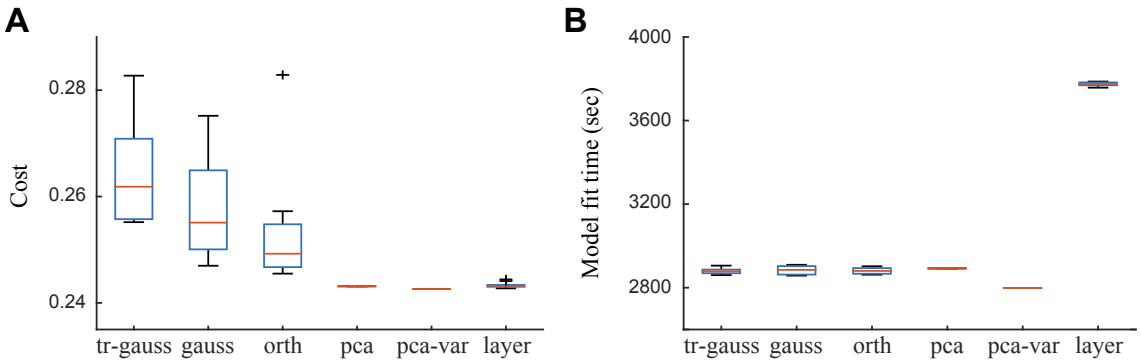

Figure 3.9: **Comparison of SRLVM initializers.** We compare different initialization schemes for a single dataset and model architecture: mouse LGN data, 377 boutons ($KS093\_run03$), three hidden layers of 30-9-30 ReLU units. This is one of the largest models we fit, and these results generalize to other architectures and datasets (data not shown). Initializers: *tr-gauss* is a truncated Gaussian; *gauss* $\sim \mathcal{N}(0, 0.1)$; *orth* uses Gram-Schmidt to orthogonalize the columns of *gauss* [192]; *pca* calculates the first M principal components of the layer's input; *pca-var* then rotates these using the varimax rotation [171]; and *layer* uses the autoencoder to initialize each layer one at a time (similar to the use of Restricted Boltzmann Machines in [193]). **A**: Cost function value; boxplots are values over ten cross-validation folds. **B**: Time in seconds to fit the model.

activity $\mathbf{r}_t$ at time t are then given by

$$\mathbf{z}_t = f_{\text{enc}}(\mathbf{y}_t) \tag{3.21}$$

$$\mathbf{r}_t = f_{\text{dec}}(\mathbf{z}_t) \tag{3.22}$$

In our models we used ReLU activation functions as the pointwise nonlinearities in all hidden layers.

We performed layer-wise initialization of the weights in the encoding network by using the varimax-rotated principal components, and initialized weights in the decoding network as transposes of those in the encoding network, which we found to consistently produce models with better cross-validation performance than other intialization schemes (figure 3.9). All SRLVMs fit in this chapter were symmetric in the encoding/decoding network architectures. The negative log-likelihood was used

as the cost function, and training was performed using L-BFGS. Regularization hyperparameters were chosen through cross-validation as described in section 3.4.2.

3.4.4 Evaluating model performance

To quantify the goodness-of-fit of the different models (GAM and SRLVM) we calculated the coefficient of determination, defined as

$$R^2 = \frac{1}{N} \sum_n \left[1 - \frac{\sum_t (y_t^n - r_t^n)^2}{\sum_t (y_t^n - \bar{y}^n)^2} \right] \tag{3.23}$$

where $\bar{y}^n$ is the average value for neuron n across all trials.

To evaluate model performance of the SRLVM we performed a version of the leave-one-out method introduced in [111]: first the full model is fit to all of the training data, and then for testing the predicted activity r_t^n at time t for neuron n is calculated as

$$r_t^n = [f_{\text{dec}}(f_{\text{enc}}(\mathbf{y}_t^{-n}))]_n \tag{3.24}$$

where $\mathbf{y}_t^{-n}$ is the population activity at time t with the value for neuron n set to zero. The prediction r_t^n is calculated in this way for each value of t and for each neuron n, and all such values are then combined for the final prediction $\{\mathbf{r}_t\}_{t=1}^T$. This procedure results in low cross-validation performance if any single neuron dominates the activity of a latent variable. The same procedure is used for GAMs that use population activity to infer latent variables (i.e. $\mathbf{x}_t^{\text{mult}} = \mathbf{y}_t$ or $\mathbf{x}_t^{\text{add}} = \mathbf{y}_t$).

Chapter 4: The Latent Variable (LV) Decoder

4.1 Introduction

An understanding of how neural activity underlies brain function requires an understanding of how individual neurons act as a population to represent information about the external world. Recent advances in recording technology now provide increasingly large numbers of simultaneously recorded neurons, and this ability is growing at an exponential pace [7]. Such observed population activity can greatly inform theories about how the brain represents information, because the manner in which neural activity is coordinated across space and time can affect how information is represented and made available to the rest of the brain [70].

Empirical evidence suggests that the structure of this coordinated neural activity is low-dimensional (section 1.5), which has implications for how information is represented in neural populations during single experimental trials. One way to access the information contained in single-trial neural responses is to decode the identity of an external stimulus using those responses [70]. Here we exploit the observed low-dimensional structure of neural activity to develop a new decoding framework that efficiently extracts information from neural populations. This framework allows us to infer and remove correlated variability that is detrimental

to decoding.

We demonstrate the use of this novel decoding framework with large populations of simultaneously recorded neurons in prefrontal cortex (PFC) and primary visual cortex (V1) of macaque monkeys. This framework leads to two advances: first, we show that a linear version of our algorithm provides an estimator for the decoded stimulus that is more efficient than other commonly-used linear estimators, using both simulated and experimental data; second, we show that this framework admits a simple extension to effective nonlinear decoding. Our nonlinear decoder extracts more information from population responses than other linear and nonlinear decoders, though the increase in complexity requires more data for fitting.

Additionally, this work contributes a conceptual advance regarding the role and consequences of correlations in neural activity. Our results demonstrate much of the noise correlations might not actually influence the representation of information, if the underlying structure of the noise is known or can be inferred by the relevant brain regions. Because a nontrivial amount of "noise" in neural activity is actually the result of non-random processes internal to the brain [105], understanding how these signals affect the representation of information could yield fundamental insights into the role of top-down control of sensory processing.

4.2 Results

4.2.1 Describing noise correlations with latent variables

We begin by describing a simple example that elucidates the relationship between signal correlations, noise correlations and latent variables which will motivate the development of our latent variable decoding framework in the following section. Consider a task in which the subject sees one of two possible stimuli on each trial and must perform a saccade to the presented stimulus, while we record the simultaneous activity of many neurons in a task-relevant brain region. To visualize how the neural dynamics unfold in time, we perform dimensionality reduction on the trial-averaged responses using principal component analysis (PCA), and project the high-dimensional activity into the first three principal components (figure 4.1A, *bold lines*). This type of dimensionality reduction is useful beyond just visualization, as many recent studies have found that a small number of appropriately chosen dimensions can explain a large fraction of the high-dimensional, trial-averaged neural activity [160, 194]. One implication of these findings is that correlations between the tuning functions of pairs of neurons, termed signal correlations, have a low-dimensional structure, a result that has been replicated across various species, brain regions, and tasks.

A full understanding of how dynamics underlie important neural computations, however, requires understanding *single*-trial activity. For example, trial-averaged trajectories reveal nothing about the differences between correct and error trials. A

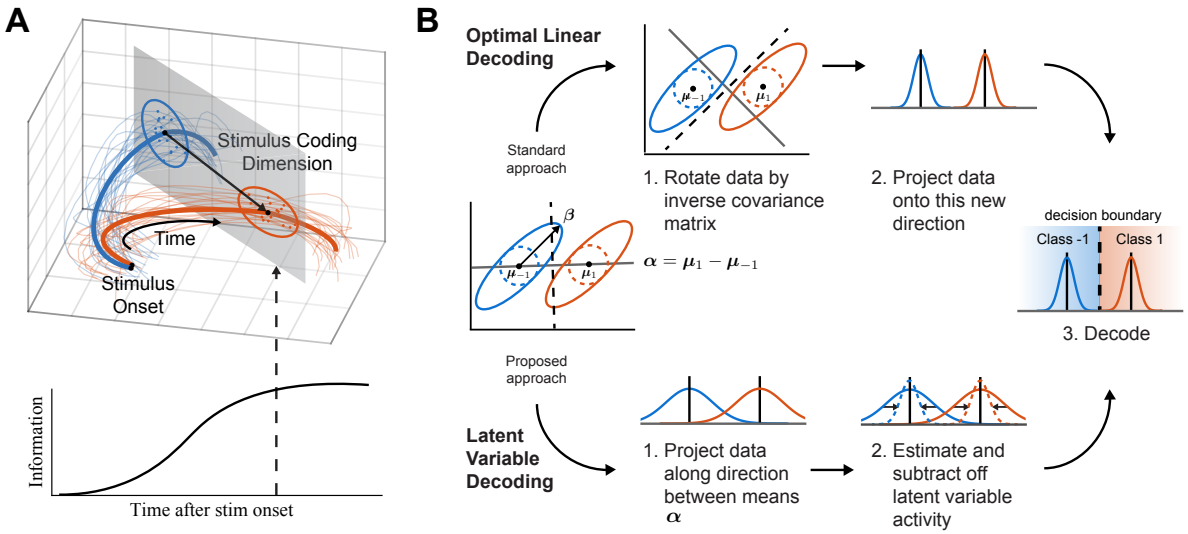

Figure 4.1: **Using latent variables to decode stimulus identity from neural population activity.** **A**: Illustration of the dynamics of high dimensional neural activity during a visually-guided saccade task, visualized using a latent variable model fit to trial-averaged activity. The trial-averaged activity at stimulus onset is marked by black dots, and evolves over time according to the saccade direction (leftward saccade, *bold blue line*; rightward saccade, *bold red line*). Activity from individual trials can also be projected into this space for visualization (*thin lines*). At a given point in time (*gray plane*), the saccade direction can be decoded from the neural activity by considering the position of the activity along the stimulus coding direction (*straight black arrow*). The more accurately the direction can be decoded from neural activity, the more information the population contains about the stimulus. **B**: Instead of using latent variables to visualize dynamics, the latent variable decoder exploits the covariance structure of the neural activity to remove variability that is shared among many neurons to improve decoding (*bottom*). This approach offers an alternative to optimal linear decoding (*top*), which accounts for the covariance structure by rotating the decoding direction by the inverse covariance matrix.

straightforward way to visualize single-trial activity is to project it into the space defined by the trial-averaged data (figure 4.1A, *thin lines*), though note that this space might not capture a large portion of the single-trial variability.

To help make the connection between noise correlations and latent variables, we consider neural activity at a single time point t during the experiment, and simplify the picture by only visualizing the neural activity in the two-dimensional

plane that passes through the mean trajectories at time t (figure 4.1A, *gray plane*; figure 4.1B, *left*). Furthermore, we assume at time t the activity of neuron n on trial i is the sum of three terms (and drop the dependence on t): 1) the response to stimulus s_i, with coupling strength α_n; 2) the response to a latent variable z_i, which is independent of the stimulus, with coupling strength β_n; and 3) a noise term ϵ_i^n, which is independent of both the stimulus and the latent variable:

$$r_i^n = \alpha_n s_i + \beta_n z_i + \epsilon_i^n \tag{4.1}$$

The latent variable z_i introduces trial-to-trial variability that is shared across the population, and could represent an internal signal such as attention or arousal [105]. The noise term ϵ_i^n, on the other hand, represents trial-to-trial variability that is private to each neuron, and could be due to mechanisms such as stochastic vesicle release [71]. If we define the variance of z_i as σ_z^2 and the variance of ϵ_i^n as σ_ϵ^2, then the covariance matrix of the full population activity $\mathbf{r}_i = [r_i^1 \dots r_i^N]^\intercal$, conditioned on the stimulus (an unscaled version of the noise correlation matrix), is given by

$$\mathrm{Cov}(\mathbf{r}_i|s_i) = \Sigma = \sigma_z^2 \boldsymbol{\beta}\boldsymbol{\beta}^\intercal + \sigma_\epsilon^2 I \tag{4.2}$$

where $\boldsymbol{\beta}$ is the vector of all β_n's. The component due to the private noise term ϵ_i^n is a scaled version of the identity matrix, and thus produces variability that is isotropic in space (figure 4.1B, *left*, *dashed circles*). The component due to the latent variable is a rank-1 matrix given by the outer product $\boldsymbol{\beta}\boldsymbol{\beta}^\intercal$, which highlights how the latent variable induces a low-dimensional structure on the noise covariance matrix. This latent variable component produces variability that is oriented along the direction $\boldsymbol{\beta}$ (figure 4.1, *left*, *solid ellipses*). This example generalizes to K latent variables,

which would produce a rank-K component in the noise covariance matrix (and point along K different dimensions in the neural response space).

4.2.2 Decoding in the presence of latent variables

We now want to understand how the presence of the latent variable affects the amount of information the neural responses contain about the identity of the stimulus on trial i. A straightforward approach for measuring linear information is to train a linear decoder to predict the stimulus using the neural activity (though see [195] for a direct estimation approach). Using the decoding approach, the optimal linear estimate of the stimulus s_i is defined as

$$\hat{s}_i = s_o + \frac{\mathbf{f}'^\intercal \Sigma^{-1}}{\mathbf{f}'^\intercal \Sigma^{-1} \mathbf{f}'} (\mathbf{r}_i - \bar{\mathbf{r}}) \tag{4.3}$$

where s_o is the average stimulus, $\mathbf{f}'$ is the vector of tuning curve derivatives with respect to s_i, and $\bar{\mathbf{r}}$ is the average population activity vector [70]. Linear Fisher information quantifies the accuracy of this estimate, and is defined as the inverse of the variance of $\hat{s}_i$ [70].

Because the noise covariance matrix Σ plays a central role in the optimal linear decoder's estimate of the stimulus, theoretical work has often addressed how decoding is affected by structure in this matrix that can arise in the neuroscience-related setting. These considerations include the impact of noise correlations that are related to the signal correlations [90], the relationship between diagonal and off-diagonal elements [92], and a noise component that points along the direction of individual neurons' tuning curves [93]. However, no one has yet explicitly considered

the consequences of a low-dimensional component arising from latent variables on decoding.

The optimal linear decoder takes the structure of Σ into account by left-multiplying $\boldsymbol{\alpha}$ by Σ^{-1} to construct a decision boundary (figure 4.1, *top*). Our latent variable decoding framework proposes an alternative approach to the optimal linear decoder by making the assumption that the noise covariance matrix contains low-dimensional structure. We outline the approach here, and more details can be found in section 4.4.1. We first project the full, high-dimensional neural activity onto the direction of $\boldsymbol{\alpha}$ (figure 4.1B, *Latent Variable Decoding step 1*), which in general is not the optimal decoding direction. Next, we form a trial-by-trial estimate of the variability in the direction of $\boldsymbol{\alpha}$. It is possible to form this estimate because of the latent variable's shared effect on the activity of the whole population; it would be impossible to form from the activity of any individual neuron, since its impact on neural activity is indistinguishable from the noise term ϵ_i^n without the single-trial statistical power gained from simultaneously recorded neurons. Next we take our estimate of this variability and subtract it from the projected population activity to reduce variability in the activity along this direction (figure 4.1B, *Latent Variable Decoding step 2*), then finally decode the adjusted neural activity by comparing to a threshold value (figure 4.1B, *Latent Variable Decoding step 3*).

Though this linear latent variable decoder cannot, by definition, outperform the optimal linear decoder in the limit of an infinite number of trials, we show in the following sections that this decoder uses data more efficiently than other linear decoders, requiring fewer trials to extract the same amount of information. An

additional feature of this framework is that it is not restricted to a linear method; indeed, using any nonlinear regression technique such as a neural network to estimate variability in the direction of $\boldsymbol{\alpha}$ will result in a nonlinear latent variable decoding algorithm, which we also explore in the following sections.

4.2.3 Validating the LV decoder with simulated data

We first tested the latent variable (LV) decoders (linear and nonlinear; see section 4.4.1 for details) on simulated data, where it was possible to compare their performance to ground truth. Responses from 200 neurons were generated in a manner similar to equation 4.1, so that the same low-dimensional covariance matrix describes the variability around each of two mean responses (figure 4.2A, *inset*; see section 4.4.3 for simulation details). This data allowed us to analytically calculate the linear Fisher information, and served as a useful test case for our method.

To evaluate the performance of the LV decoders on this data we estimated Fisher information by calculating d'^2 in the learned decoding direction (equation 4.12), and compared this to the true linear Fisher information (equation 4.15). d'^2 is a quantity that must be estimated from data, and even the optimal linear estimator cannot extract the full linear Fisher information from limited data. As the number of neurons that can be simultaneously recorded increases exponentially, the limiting factor for accurately measuring information in population activity will be the number of experimental trials.

We tested the LV decoders across a range of trial-to-neuron ratios (figure 4.2).

Both the Linear and Nonlinear LV decoders extracted a large fraction of the true linear information using relatively few trials (figure 4.2A). Because this data does not contain nonlinear information, the Nonlinear LV decoder cannot perform better than the Linear LV decoder. We also show the performance of the "Difference of means" decoder (4.2A, *purple dashed line*), which uses the same decoding direction as the LV decoders, but does not estimate and subtract off variance. The gap in performance between the Difference of means and LV decoders demonstrates the extent to which the LV decoders are able to account for shared variability that is detrimental to decoding (4.2B). [See figure 4.8 for a comparison between the Linear LV decoder and other standard linear decoders on this simulated data.]

To gain some intuition about how the neural networks might learn to predict the variability that is due to the latent variables, we analytically work out a simple example in section 4.5. This example suggests that the Linear LV decoder should project the high-dimensional neural activity onto a direction orthogonal to the stimulus coding dimension $\boldsymbol{\alpha}$ to produce an estimate of the variability. To test this, we projected the $\boldsymbol{\alpha}$ direction out of the data and retrained the Linear LV decoder, and found that its performance did not change (figure 4.2C).

The amount of information that a linear decoder can extract from data is bounded above by the linear Fisher information. However, real data may contain nonlinear Fisher information that requires nonlinear decoding techniques to extract. In some cases, this nonlinear component of the full Fisher information can be substantially larger than the linear component [91]. To date, however, there are very few examples of the successful application of nonlinear decoders to neural data,

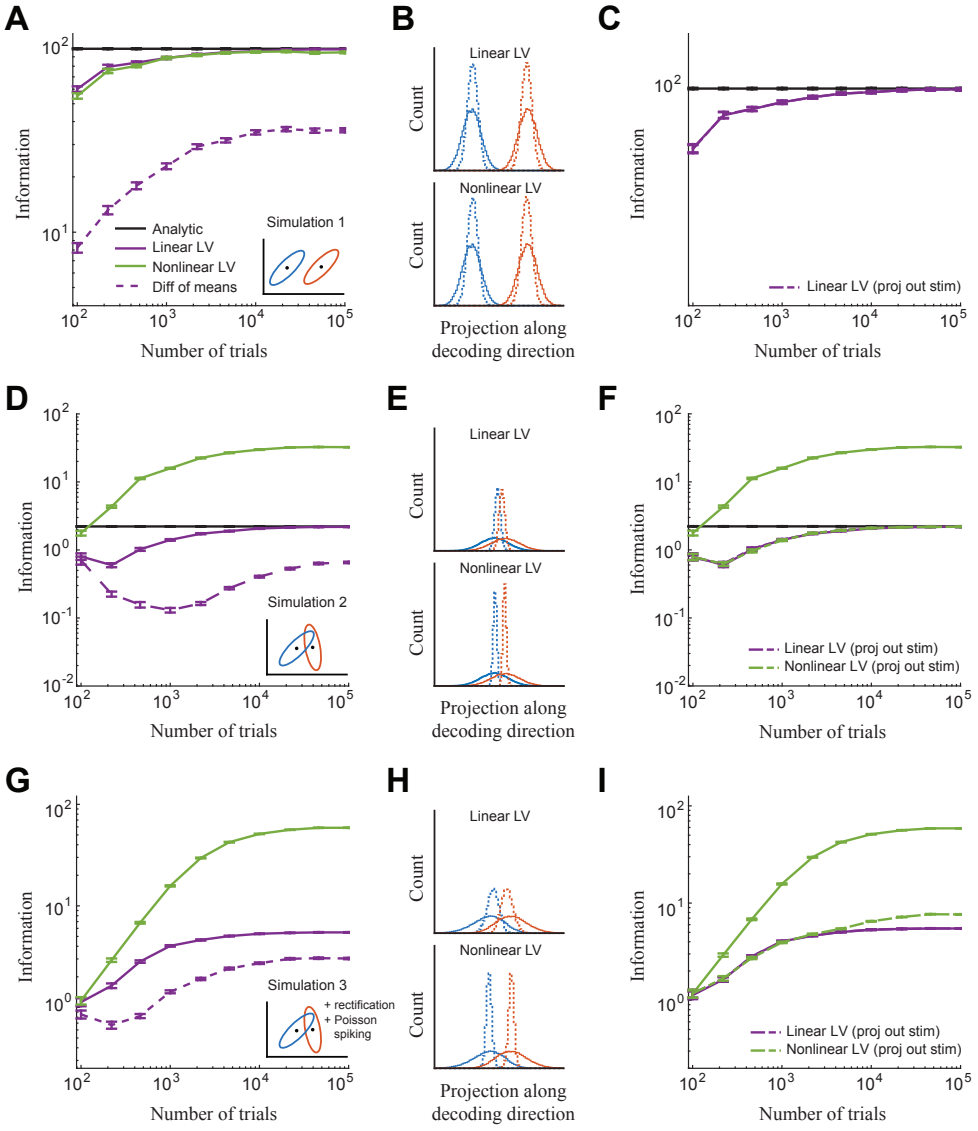

Figure 4.2: **Linear and Nonlinear LV decoding performance on simulated data.** **A**: Gaussian data is generated using the same covariance matrix for each class (*inset*), which only contains linear Fisher information (*black line*). Information measure is the d'^2 discriminability index described in section 4.4.2, and error bars represent SEM over five cross-validation folds for each of 25 simulated datasets. **B**: Histograms of the data projected onto the decoding direction for one dataset, colored by class, both before (*solid lines*) and after (*dashed lines*) subtracting off the predicted variability using the Linear LV (*top*) or Nonlinear LV (*bottom*) decoders. **C**: Decoder performance when projecting the stimulus dimension $\boldsymbol{\alpha}$ out of the population activity before using it to infer variability. **D**–**F**: Same as *A*-*C*, but data is generated using a different covariance matrix for each class (see inset in *D*). **G**–**I**: Same as *D*-*F*, but the resulting values are rectified and passed through a Poisson spike generator to simulate spike count data.

largely because these methods require much more data than their linear counterparts (see [196] for the application of nonlinear decoders to neural data in a different setting).

To demonstrate the performance of our Nonlinear LV decoder, we introduce a new simulation wherein the noise covariance matrices are different for each mean response (4.2D, *inset*). This simulation explicitly models stimulus-dependent noise correlations, which have been observed in neural data [77, 78]. Data generated in this way contains nonlinear information [92], unlike the previous simulation, and thus serves as a natural extension for testing nonlinear decoders.

Figure 4.2D-F (analogous to 4.2A-C) demonstrates that the Nonlinear LV decoder can extract nonlinear Fisher information that is orders of magnitude larger than the linear Fisher information, and can do so with even a small number of trials. It is interesting to note that the Nonlinear LV decoder takes a different strategy for inferring the variability; when we projected $\boldsymbol{\alpha}$ out of the data before fitting the Nonlinear LV decoder its performance decreased substantially, becoming equivalent to that of the Linear LV decoder (4.2F). This points to the ability of the Nonlinear LV decoder to extract nonlinear information by explicitly using activity in the direction of $\boldsymbol{\alpha}$.

The simulated data analyzed above still lacks several important statistical features of neural data. To ensure that the Linear and Nonlinear LV decoders still perform well in a more realistic setting, we simulated data that contained stimulus-dependent noise correlations, then rectified the responses and fed them into a Poisson spike generator to mimic the discrete nature of spiking data (4.2G-I). Both LV

decoders perform similarly to the simulation in figure 4.2D-F, demonstrating the ability of these techniques to generalize well to non-Gaussian data.

4.2.4 Decoding PFC activity during decision-making

We now turn to the analysis of experimentally-measured neural activity to test the performance of our LV decoding framework on real data. We first analyzed data recorded during a visually-guided saccade task that resembles the structure of the simulated data (see section 4.4.4). Briefly, two macaque monkeys were trained to fixate on a central point, after which a saccade target appeared either to the left or right of the central point. The subject then had to saccade to the target and hold fixation for 500 ms before receiving a stochastic reward on correct trials. During the task, neural activity was recorded from eight bilaterally-implanted Utah arrays in the prefrontal cortex (four per hemisphere), resulting in ∼600 simultaneously recorded single- and multi-units from each subject over ∼2000 trials. We analyzed spike counts in the 500 ms fixation period following the saccade.

This dataset provides a unique opportunity to study the performance of decoders in many different regimes due to the large number of neurons and trials. We first looked at how linear decoder performance scaled with the population size, to sample a range of trial-to-neuron ratios (figure 4.3). The Linear LV decoder was able to extract more information than the other methods across all population sizes in both monkeys. The information extracted by all linear decoders begins to saturate with increasing population size, though the asymptotic value depends on the

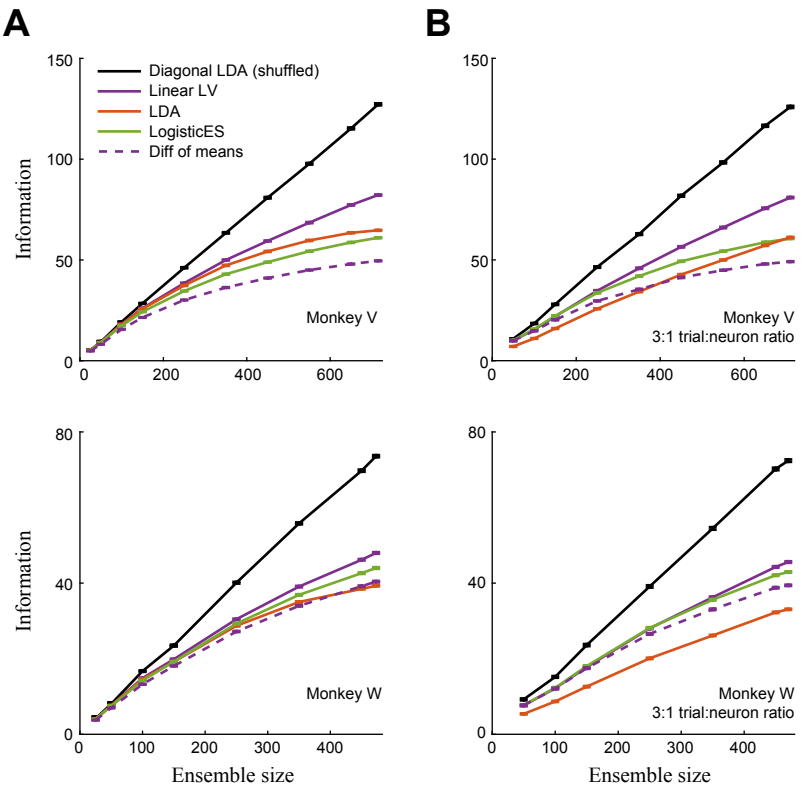

Figure 4.3: **Linear LV decoder extracts more information than standard linear decoders.** **A**: The estimated information extracted by various decoders as a function of ensemble size for two monkeys (*top* and *bottom*). In both monkeys, the Linear LV decoder outperforms the other tested methods. The solid black line indicates the information extracted by the optimal linear decoder on a shuffled version of each dataset (Diagonal LDA), which illustrates the extent to which correlations negatively impact information coding. The saturating behavior of all linear decoders (on unshuffled data) suggests the presence of information-limiting noise correlations in the data. Error bars represent SEM over ten cross-validation folds and 100 independently sampled ensembles from the data (see section 4.4.4). **B**: Fixing the number of trials used to train the decoders to be three times the ensemble size changes the scaling behavior, demonstrating that the saturation of curves in A is due in large part to data limitations (see table 4.1). The Linear LV decoder performs well even in the low-trial, high-neuron regime common to many modern electrophysiology experiments.

decoding algorithm (table 4.1). This saturation could be due to either the presence of information-limiting noise correlations [93], or suboptimal decoders [195], or both.

To understand the extent to which noise correlations (information-limiting

		Linear LV	LogisticES	LDA	Diff of means
Monkey V	All data	**202**	103	112	77
	3:1 ratio	**283**	110	240	78
Monkey W	All data	**129**	103	74	91
	3:1 ratio	**160**	117	128	93

Table 4.1: **Information saturation estimates are affected by estimators and data limitations.** The value at which information saturates for infinite population size is estimated for the curves in figure 4.3 (see section 4.4.2). This saturation value is highly dependent on the type of decoder used, as well as the amount of data used for training the decoder.

and otherwise) affect the amount of information encoded by the population, for each neuron we shuffled the trials within each stimulus condition to remove noise correlations [92]. We then used a factorized decoder (Diagonal LDA; see section 4.4.2) that explicitly ignores correlations to extract information from the resulting shuffled population responses, which does not saturate in this regime (figure 4.3A, *black line*). Because information in the shuffled data was larger, the structure of the noise correlations in the original data could be detrimental to information encoding in this population, a point we return to in the discussion.

We also assessed how the saturation values of the curves in figure 4.3A are affected by lack of training data (i.e. suboptimal decoding) by looking at their performance while holding the trial:neuron ratio fixed (figure 4.3B). This allowed for a fairer comparison of the information measure across ensemble sizes. We fit a saturating function to these curves to estimate their asymptotic values (see section 4.4.2). Indeed, the saturation value of these lines increased due to a decrease in

performance for smaller ensemble sizes, which previously had access to a larger amount of data than the larger ensemble sizes (table 4.1). This result demonstrates the need to interpret these plots carefully - though saturation behavior still exists for the linear decoders, the true saturation point is highly dependent on both the estimator and the amount of training data.

We next considered the performance of Nonlinear LV decoder on this experimental data while constraining the trial:neuron ratio to be 10:1 (figure 4.4A; note that this constrains the maximum ensemble size to be $\sim$200 neurons). The Nonlinear LV decoder extracted significantly more information from the population responses than even the factorized decoder on shuffled data. As an additional control, we also compared to the direct estimator of linear Fisher information developed in [195] (figure 4.4A, *dashed black line*). The superior performance of the Nonlinear LV decoder only emerges for relatively large (>100) ensemble sizes, before which it is mostly equivalent to the linear estimators.

The superior performance of the Nonlinear LV decoder depends not just on ensemble size, but also the amount of data available to train the decoder. If the trial:neuron ratio is reduced to 3:1, its performance drops considerably, revealing the large data requirements of this decoder (figure 4.4B; note the change in x- and y-axis bounds). To demonstrate the power of the Nonlinear LV decoder in a non-data-limited regime, we fixed the ensemble size to be 25 and trained the Nonlinear LV decoder using a range of trial numbers (figure 4.4C). Its performance steadily increases with the amount of training data and far outperforms the linear estimators (note that the direct estimator tends to have a positive bias in the limited data

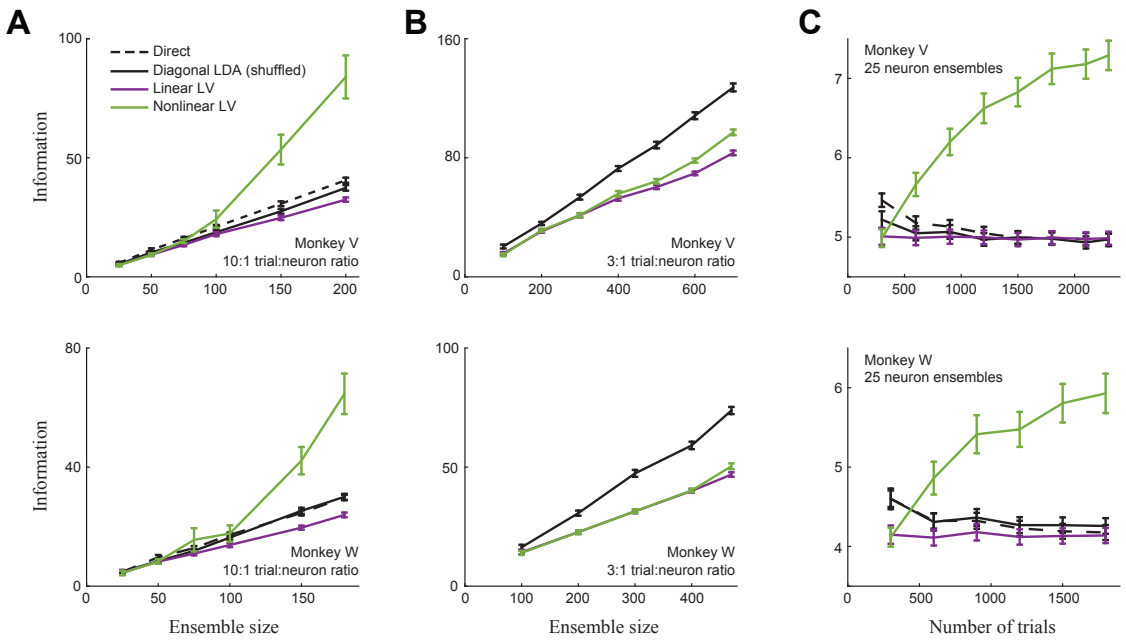

Figure 4.4: **Nonlinear LV decoder extracts more information than all linear decoders with sufficient data.** **A**: Information versus ensemble size using a 10:1 trial:neuron ratio. The Nonlinear LV decoder extracts more information than both the optimal linear decoder on shuffled data and the direct estimate of linear Fisher information (*dashed black line*; [195]) for both monkeys. **B**: The Nonlinear LV decoder performance decreases relative to the shuffled data when training the decoders using a 3:1 trial:neuron ratio. **C**: Fitting the Nonlinear LV decoder to ensembles of 25 neurons across a range of trials demonstrates that large trial:neuron ratios are needed before performance plateaus. The direct method overestimates linear Fisher information for small trial:neuron ratios, and as such is omitted from *B*. Error bars for all plots are the same as in figure 4.3.

regime, and thus we omit this curve from figure 4.4B).

4.2.5 Decoding V1 activity during passive viewing

To test if the results from the previous section generalize to other brain regions, we analyzed activity from primary visual cortex (V1) of anesthetized macaques in response to repeated presentations of 12 equally-spaced drifting gratings [191]. We decoded drift direction for each pair of gratings separated by 30° using spike counts

in a 500 ms bin during the middle of the trial (see section 4.4.4).

The Nonlinear LV decoder was again able to extract more information than the linear estimators across all three monkeys using a 10:1 trial:neuron ratio (figure 4.5A). As in the PFC dataset, this performance decreases considerably when limiting the trial:neuron ratio to 3:1 (figure 4.5B), though in all three monkeys the performance remains much better than that of the Linear LV decoder.

We also asked how well the LV decoders compared to other standard decoders on this V1 dataset, both linear and nonlinear. The Linear LV decoder extracted significantly more information than LDA and LogisticES, and the Nonlinear LV decoder extracted significantly more information than all linear estimators (excluding shuffled) for all monkeys (figure 4.5C, *top panel* for each monkey; paired t-test; $p<$1e-5 for all comparisons).

Our measure of information (section 4.4.2) is not applicable to most nonlinear decoders because they do not project the data onto a discriminant line. To compare the Nonlinear LV decoder to other nonlinear decoders [Quadratic Discriminant Analysis (QDA) and Kernel SVM using radial basis functions; see section 4.4.2 for fitting details], we used the fraction of correctly classified stimuli as a measure of performance (figure 4.5C, *bottom panel* for each monkey) [Note that in the PFC data fraction correct was at or close to one for all methods, rendering this performance measure meaningless for their comparison.] Across all three monkeys QDA and Kernel SVM actually performed worse than even the linear methods. The poor performance of QDA and Kernel SVM could be due to several factors, including: 1) the number of trials is not sufficient to expose the full power of the decoder (i.e. it is

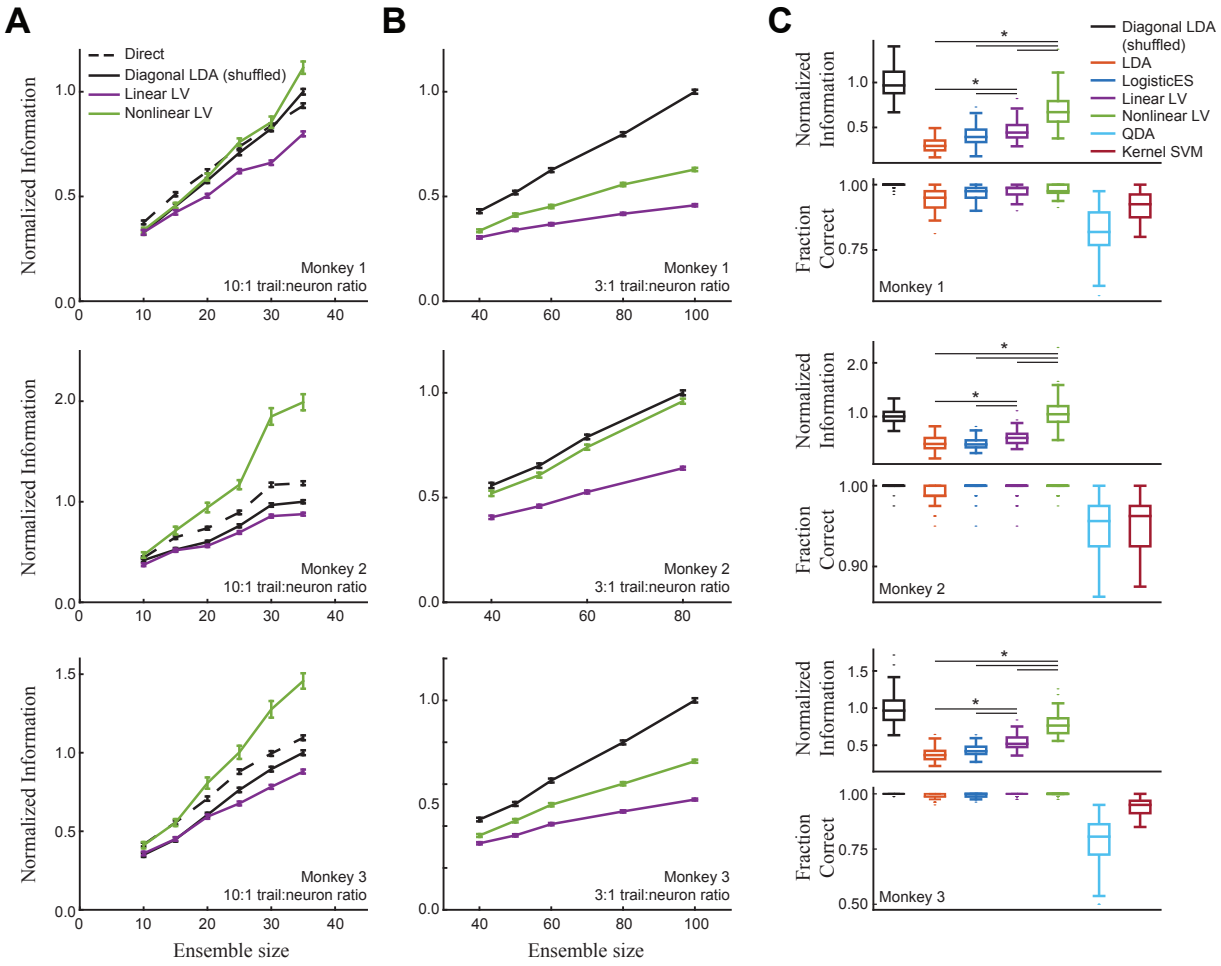

Figure 4.5: **LV decoder performance on V1 dataset.** The LV decoder performs well on data from a different brain region during a different experiment. The information measure is averaged over multiple pairwise discriminations of drifting gratings (0° vs 30°, 30° vs 60°, etc.) for each monkey by normalizing by the mean value of Diagonal LDA on shuffled data for the largest ensemble size. Error bars represent SEM over one cross-validation fold, 50 independently sampled ensembles and 12 pairwise discriminations. **A**: The Nonlinear LV decoder outperforms the Linear LV decoder using a 10:1 trial:neuron ratio. **B**: The Nonlinear LV decoder performance decreases as the trial:neuron ratio decreases to 3:1, as in the PFC data (figure 4.4). **C**: Comparison of linear and nonlinear decoder performance using the full dataset (∼4:1 trial:neuron ratio). Normalized information (*top*) is calculated as in *A* and *B*, and cannot be calculated for Quadratic Discriminant Analysis (QDA) or Kernel SVM. To directly compare these methods, Fraction Correct (*bottom*) is plotted for all decoders. The asterisk (*top*) indicates p<1e-5 (paired t-test); only comparisons with the LV decoders are shown, and comparisons to Diagonal LDA are omitted.

an inefficient estimator); and 2) the nonlinear decision boundaries that the decoder is capable of learning are not well-matched to those found in the data (which is more likely with QDA, which can only learn quadratic decision boundaries). We leave this as an open question for future work.

4.3 Discussion

We presented a novel framework for decoding neural activity that is motivated by the observation that noise correlations in large populations of neurons can often be explained by a small number of latent variables (figure 4.1). We infer the variability due to these latent variables along a chosen decoding direction through their influence on the population activity, and remove this variability to reduce its detrimental effect on decoding. We showed that a linear version of this latent variable decoder is more efficient in its use of data than other common linear decoders, using simulated data (figure 4.8) and experimental data from PFC (figure 4.3) and V1 (figure 4.5). We also used this framework to develop a nonlinear latent variable decoder, and showed that it can extract nonlinear information in simulated (figure 4.2) and experimental data (figures 4.4 and 4.5).

Decoding neural activity is a conceptually straightforward way to estimate the amount of information a neural population contains about an external stimulus. However, the performance of decoders can be greatly limited by low trial:neuron ratios (figures 4.4 and 4.5), which may affect the conclusions that can be drawn from these types of analyses, and highlights the need to develop more efficient decoding

techniques such as the Linear LV decoder. This problem will continue to grow as recording technologies allow experimenters to simultaneously record from increasingly large populations of neurons, without a concomitant increase in the number of trials (though see [197]).

4.3.1 Information-limiting noise correlations

Theoretical work has uncovered the particular structure of noise correlations that limit the information represented by populations of neurons, termed *information-limiting* or *differential* noise correlations [93]. These correlations point directly along the stimulus coding direction $\boldsymbol{\alpha}$ (or along the derivative of the tuning curve for continuous-valued stimuli), so that the noise is indistinguishable from a change in the signal.

The presence of these information-limiting noise correlations is difficult to measure directly from experimental data, because they can be extremely small and masked by other patterns of correlations [93, 94, 195]. Nevertheless, their existence in neural activity is certain (at least with respect to sensory information) due to the data processing inequality, which states that no transformations can increase the amount of information available at the sensory receptors [94, 198]. Because sensory receptors are themselves noisy [71], information about the sensory periphery contained in neural activity must therefore be limited.

That fact that sensory information is limited does not, however, obviate the need for complex, nonlinear transformations of representations in neural systems.

Indeed, all the information needed to correctly discriminate images of cats and dogs is available in the activity of retinal neurons, but a series of nonlinear transformations is needed to make that information linearly accessible [199]. Nonlinear transformations can also, for example, lead to sparse representations that are robust to noise [76] and energetically efficient [31].

Our decoding framework, though designed to reduce variability in the coding direction, does *not* remove information-limiting correlations (and indeed cannot, by definition). It is important to note that information-limiting correlations must be exactly aligned with the coding direction ($\boldsymbol{\alpha}$ in figure 4.1B). Instead, our decoder removes variability that points in other directions of neural activity space (*non-information-limiting noise correlations*), but has a non-zero projection into the coding direction (the projection of $\boldsymbol{\beta}$ onto $\boldsymbol{\alpha}$ in figure 4.1B).

4.3.2 Non-information-limiting noise correlations

If information-limiting noise correlations are an inevitable byproduct of the data processing inequality, then how do the non-information-limiting correlations arise? The sensitivity of noise correlations to behavioral context [79, 80], attention [86–88] and perceptual learning [84, 86] suggest that they are at least partially the product of top-down, feedback processes in sensory cortex. However, these types of correlations can reduce the amount of information available in a neural population of finite size [92] (though not limit the information as the population becomes infinitely large [93]).

What then is the functional role of these correlations? Theoretical work has shown how inducing particular patterns of noise correlations improves information transmission between different brain regions [181]. Another line of theoretical inquiry implicates correlated variability in the representation of prior information in a Bayesian framework of sensory integration [105, 200]. Here we show that the goals of robust information propagation or perceptual inference (where noise correlations can help) and accurate decoding (where noise correlations can hurt) are not necessarily at odds with one another. We hypothesize that if activity in a brain region is corrupted by latent-variable-induced noise correlations, a downstream decoder of that activity need not be negatively impacted if it has access to the top-down signal inducing the correlations (a possibility also addressed by [105] and [80]). Decision-making areas could then conceivably integrate sensory information with these top-down signals, an idea that has recently been explored in modeling work [200, 201], and was shown to explain a variety of empirically-observed phenomena related to noise correlations.

Our results are also consistent with a recent study demonstrating that the inclusion of activity from untuned neurons can increase the performance of decoders [182]. In our framework, these neurons might not be tuned to the particular task, but they can still carry information about the signals that give rise to correlated variability. Including these neurons in a decoder can then lead to more accurate estimation of the trial-to-trial variability, which in turn will improve the performance of the decoder.

4.3.3 Limitations of the study

We demonstrated that our latent variable decoding framework extracts more information from neural responses than other standard decoders, both linear and nonlinear. However, this comparison was based on an approximation of linear Fisher information because all decoders tested could achieve perfect performance on the fraction of correctly classified trials using the full population (and indeed, much smaller ensembles as well). If the animal is able to perform the correct behavior 100% of the time, is an increase in information with larger population sizes even meaningful? Is the saturation of information with increasing population size a relevant problem for neural information processing?

These concerns also bring into question the usefulness of our nonlinear decoder. Again, is it meaningful to extract more information from neural responses when the linear decoders already correctly predict the stimulus on all held-out trials? Furthermore, just because nonlinear information is present in a given brain region does not mean that the brain is able to use it (and this nonlinear information must saturate as well due to the data processing inequality). Returning to our cat and dog example in the retina, all of the information about object identity is present at that first stage of visual processing, but is presumably not utilized until much further along the visual processing hierarchy. Despite these concerns, our Nonlinear LV decoder is relatively general, with an intuitive explanation for the computation it performs, and thus could conceivably be implemented in a single stage of neural processing.

We must also consider the fact that we are using an extremely simple binary choice task that employs an extremely simple cue, so that the decision-making task is one-dimensional. Of course, ethologically-relevant situations will typically contain much higher-dimensional stimuli (with many information-limiting directions) and higher-dimensional decision spaces. However, as we show here, accurate estimates of information require large numbers of trials, even with our efficient decoding framework. Future work will have to focus on richer experimental paradigms in order to understand the extent to which the animal actually makes use of the information we are now able to extract [202].

4.4 Methods

4.4.1 The LV decoder

Training the decoder. For a population of N neurons recorded during T trials, we define $R = [\mathbf{r}_1 \dots \mathbf{r}_T]^\intercal \in \mathbb{R}^{T \times N}$ to be the matrix of spike counts and $\mathbf{y} = [y_1 \dots y_T]^\intercal \in \{\pm 1\}^T$ to be a binary vector that indicates the stimulus identity $s \in \{\pm 1\}$ on each trial $i \in \{1, \dots, T\}$. The following steps are illustrated in figure 4.6.

Step 1: We first estimate the mean stimulus responses $\hat{\boldsymbol{\mu}}_s$ as

$$\hat{\boldsymbol{\mu}}_s = \frac{1}{N_s} \sum_{\{i | y_i = s\}} \mathbf{r}_i \tag{4.4}$$

where N_s is the number of stimuli from class s. We then estimate the stimulus coding direction $\hat{\boldsymbol{\alpha}} \in \mathbb{R}^N$ that points between the two mean stimulus responses as

$\hat{\boldsymbol{\alpha}} = \hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_{-1}$.

Step 2: Next, we project R onto $\hat{\boldsymbol{\alpha}}$ to reduce the high-dimensional neural activity into a single dimension,

$$\mathbf{r}^{\alpha} = R\hat{\boldsymbol{\alpha}} \tag{4.5}$$

which is now scalar value for each trial i. Directly decoding $\mathbf{r}^{\alpha}$ results in the difference of means decoder used throughout the text. This decoder generally performs much worse than the latent variable decoder, even though both use the same decoding direction, and demonstrates the extent to which variability in the direction of $\mathbf{r}^{\alpha}$ is both 1) detrimental to decoding, and 2) shared across multiple neurons, and hence inferrable from the data. Note that in this context the locally optimal decoding direction is defined as $\mathbf{r}^{\text{opt}} = \hat{\Sigma}^{-1}\hat{\boldsymbol{\alpha}}$, where $\hat{\Sigma} = \frac{1}{2}(\hat{\Sigma}_{-1} + \hat{\Sigma}_1)$ is the average of the noise covariance matrices estimated separately for each stimulus condition [70].

Step 3: Variability in the vector $\mathbf{r}^{\alpha}$ is due to both the stimulus response and to "noise", though we only want to estimate the noise component. To do so, for each trial i we subtract the appropriate mean stimulus response from the projection along $\hat{\boldsymbol{\alpha}}$ and denote this new quantity $\mathbf{r}^{z}$:

$$(\mathbf{r}^{z})_i = (\mathbf{r}^{\alpha})_i - \hat{\boldsymbol{\alpha}}^{\intercal}\hat{\boldsymbol{\mu}}_{y_i} \tag{4.6}$$

where $(\mathbf{x})_i$ denotes the ith component of a vector $\mathbf{x}$. The extent to which $\mathbf{r}^{z}$ contains variability that is shared across many neurons (due to latent variables) determines the efficacy of the latent variable decoder.

Step 4: We estimate this shared component of variability with the full population response R by learning a mapping $f_\theta : \mathbb{R}^N \to \mathbb{R}$ parametrized by θ using a

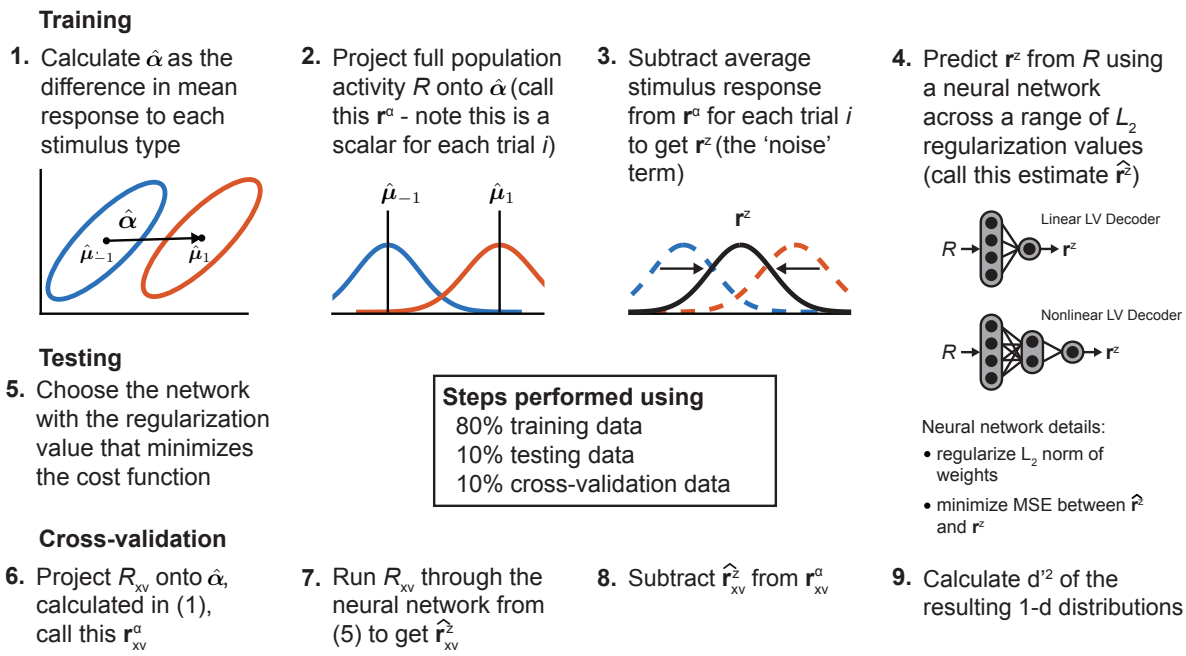

Figure 4.6: **Outline of LV decoding algorithm.**

neural network (see *Neural network details* below), so that

$$(\hat{\mathbf{r}}^{z})_i = f_\theta(\mathbf{r}_i) \tag{4.7}$$

Evaluating the decoder. Once f_θ has been learned, we can evaluate the performance of the latent variable decoder. The following steps can be performed on any subset of the data since no parameters are being learned. As an example, we will use R_{xv} to represent the cross-validation data. R_{xv} is first projected onto $\hat{\boldsymbol{\alpha}}$ (learned from the training data) to get $\mathbf{r}^{\alpha}_{\text{xv}}$ (*step 6*). The activity $\mathbf{r}_{\text{xv}}$ from each trial is then run through the function f_θ to produce one component of the vector $\hat{\mathbf{r}}^{z}_{\text{xv}}$ (*step 7*). Finally, the variance-reduced activity is given by (*step 8*)

$$\tilde{\mathbf{r}}_{\text{xv}} = \mathbf{r}^{\alpha}_{\text{xv}} - \hat{\mathbf{r}}^{z}_{\text{xv}} \tag{4.8}$$

To calculate the classification of each trial i, the corresponding value from $\tilde{\mathbf{r}}_{\text{xv}}$ is

compared with $\bar{\mu} = \frac{1}{2}(\mu_1 + \mu_{-1})$; values larger than this quantity are classified as stimulus 1, and all others are classified as stimulus -1 (*step 8*).

Neural network details. Any technique that can learn a mapping from $\mathbb{R}^N$ to $\mathbb{R}$ is suitable in principle, and we use a standard neural network to do so. The neural network takes R as input and produces an estimate $\hat{\mathbf{r}}^z$ of $\mathbf{r}^z$. Parameters of the network θ are learned by minimizing the mean square error (MSE) between $\hat{\mathbf{r}}^z$ and $\mathbf{r}^z$. L_2 regularization is included to prevent overfitting to the training data [17], so that the penalized cost function C is defined as:

$$C(\theta) = \|\mathbf{r}^z - \hat{\mathbf{r}}^z\|_2^2 + \lambda\|\theta\|_2^2 \tag{4.9}$$

where $\|\mathbf{x}\|_2 = \sum_k x_k^2$ is the L_2 norm of a vector $\mathbf{x}$ and λ is a hyperparameter that controls the magnitude of the regularization term. In practice, we fit the latent variable decoders using 10 different values of λ logarithmically spaced between $1e-4$ and $1e1$, and choose the value that results in the smallest cost function when evaluated on the testing data (*step 5*). The cost function is optimized using an L-BFGS routine [177].

The linear latent variable decoder requires a linear mapping from $\mathbb{R}^N$ to $\mathbb{R}$, and therefore uses a neural network with just an input layer and an output layer. With the L_2 regularization, this network is equivalent to regularized linear regression, or "ridge regression" [17]. The nonlinear latent variable decoders use a neural network with a single hidden layer composed of rectified linear units (ReLUs), which we found to work well for both simulated and experimental data. However, many details of the network can be changed, including different numbers and types of hidden units,

more hidden layers, etc.

Projecting out the stimulus coding dimension. To test the extent to which the latent variable decoders require information contained in the stimulus coding direction $\boldsymbol{\alpha}$ (figure 4.2C, F, I), we projected this dimension out of the population activity R before using it to predict variability in the same dimension $\boldsymbol{\alpha}$, and we denote the resulting activity by $\tilde{R}$. The stimulus coding dimension was calculated after subsampling trials and neurons, and was only calculated using training data. Training the decoder then amounted to replacing R in equation 4.5 with $\tilde{R}$. To evaluate the decoder, the same $\boldsymbol{\alpha}$ was projected out of the testing/cross-validation data, and all other steps in the *Evaluating the decoder* section remain the same.

4.4.2 Training and evaluating decoders

All decoders were fit using 5-fold (simulated data) or 10-fold (experimental data) nested cross-validation, where one fold was used for cross-validation, one fold was used for testing, and the remaining folds were used for training. Each fold was used once for cross-validation. For each nested cross-validation (i.e. for a given number of neurons and trials), folds were created once by randomly sampling the available trials, and the same folds were used with each decoder. Unless otherwise noted below, all quantities were calculated using training data. Some decoders (DoM, LDA, QDA) do not have hyperparameters, and thus did not use the testing data at all.

Difference of means (DoM). The mean response to each stimulus was

calculated; the difference in mean responses $\boldsymbol{\alpha}$ defined the discriminant line for the DoM decoder, and the mean of the mean responses defined the threshold. For each trial, neural activity was projected onto the discriminant line and compared to the threshold value to determine its classification.

Linear discriminant analysis (LDA). LDA was performed using the *fitcdiscr* function in MATLAB, with the 'DiscrimType' option set to 'linear' so that a single pooled covariance matrix was estimated from the data. The 'Gamma' option was set to 0, so that the estimated covariance matrix was not regularized with an additional diagonal matrix. This choice limited the use of LDA to settings where the number of trials was larger than the number of neurons. Diagonal LDA was performed on trial-shuffled data using the same function, with the 'DiscrimType' option set to 'diaglinear' so that a single pooled, diagonal covariance matrix was estimated from the data.

Quadratic discriminant analysis (QDA). QDA was performed using the *fitcdiscr* function in MATLAB, with the 'DiscrimType' option set to 'pseudoquadratic' so that a covariance matrix was estimated for each stimulus type. The covariance matrices were inverted using the pseudoinverse, and QDA was therefore not limited by the number of trials.

Kernel support vector machine (Kernel SVM). Kernel SVMs were fit using the *fitcsvm* function in MATLAB with the 'KernelFunction' option set to 'rbf' to use radial basis function kernels. Radial basis functions are unnormalized Gaussians, and the scale of these functions relative to the data is important for kernel SVM performance. MATLAB provides another option 'KernelScale' that scales the

data (rather than the kernel); to fit this hyperparameter, we fit kernel SVMs using 10 different values of the scale parameter logarithmically spaced between $1e-3$ and $1e3$, and chose the scale that resulted in the largest number of correctly classified trials when evaluated on the testing data (see *Evaluating decoders* below).

Logistic regression with early stopping (LogisticES). Logistic regression models were fit by minimizing the mean square error between class labels $y \in \{0, 1\}$ and predicted class labels given by

$$\hat{y} = \frac{1}{1 + \exp(-R\mathbf{b} + c)} \tag{4.10}$$

where R is the matrix of neural responses, $\mathbf{b}$ is the vector of learned decoder weights and c is a learned bias term. The negative log-likelihood of the testing data was evaluated on each iteration, and model fitting terminated once the negative log-likelihood began to increase or the algorithm reached 1000 iterations [203].

Evaluating decoders. The simplest measure for evaluating decoder performance is the fraction of correctly classified trials. For LDA, QDA and kernel SVM, the predicted classification for each trial was obtained using the *predict* function in MATLAB. The DoM and latent variable decoders explicitly define a threshold, and a trial is classified based on comparing the projection of the data along the learned discriminant line to the threshold. For LogisticES, the predicted class label $\hat{y}$ (equation 4.10), a continuous quantity between 0 and 1, was turned into a binary classification by using 0.5 as a threshold.

The fraction of correctly classified trials, or accuracy A, can be directly converted to the d' measure using the inverse of the complimentary error function

H [92]:

$$d'_{\mathrm{FC}} = 2H^{-1}(1 - A) \tag{4.11}$$

where 'FC' denotes 'fraction correct'.

Whenever classes are fully separated, however, d'_{FC} is not an adequate measure of information. Linear Fisher information measures the inverse of the variance of a decoder's prediction of the stimulus, and therefore decoders with smaller variance in their predictions should contain more information. However, if two decoders are able to correctly classify all trials, then d'_{FC} is unable to distinguish between a decoder with high variance and one with low variance (as long as $A = 1$). Therefore, to study information in population activity in this regime (e.g. figure 4.3), d' is calculated by estimating the number of correctly classified trials in the limit of infinite data, and denoted d'_{MLE}.

To calculate the d'_{MLE} measure, the full-dimensional population activity is first projected onto the discriminant line (which precludes the use of this measure with QDA or kernel SVM, which do not estimate discriminant lines). The resulting one-dimensional projection for each class is well-described by a Gaussian distribution (data not shown). The mean and variance of this distribution is fit for each class using the maximum likelihood estimates. Then, the fraction of correctly classified trials for each class, in the limit of infinite data, is estimated by using the error function. For example, if the mean of class 0 is located to the left of the threshold, the number of correctly classified trials is given by the area under the curve between the threshold and negative infinity, and is denoted by A_0 (A_1 is defined analogously

for the other class). The fraction of correctly classified trials is then estimated as $\hat{A} = \frac{1}{2}(A_0 + A_1)$, and

$$d'_{\text{MLE}} = 2H^{-1}(1 - \hat{A}) \tag{4.12}$$

Equivalence of linear Fisher information and d'^2. Throughout this chapter, we refer to $(d'_{\text{MLE}})^2$ as 'Information'. To justify this equivalence, we show here that yet another definition of d' is equivalent to linear Fisher information when calculated along the optimal coding direction and squared. Though these three values of d' differ in their computation, under the assumption of Gaussianity they become equivalent in the limit of infinite data. We now consider the classic definition of d' [204], which was originally introduced in the signal detection literature as a measure of the signal-to-noise ratio (SNR):

$$d'_{\text{SNR}} = \frac{\mu_1 - \mu_0}{\sigma} \tag{4.13}$$

where μ_i is the mean of the ith one-dimensional response distribution and σ is the standard deviation, which we take to be the same for both distributions. d' decreases as the distance between the distribution decreases, and also decreases when the distance is fixed but the standard deviation increases. If we now consider the response $\mathbf{r}$ of a population of neurons, with a stimulus-conditioned covariance matrix given by $\text{Cov}(\mathbf{r}|s) = \Sigma$, the definition of linear Fisher information in this context becomes

$$I = \mathbf{f}'^{\intercal}\Sigma^{-1}\mathbf{f}' \tag{4.14}$$

$$= (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^{\intercal}\Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) \tag{4.15}$$

We now calculate d'^2_{SNR} of the response distributions after they have been projected along the optimal decoding direction $\mathbf{w}^\intercal = \frac{(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^\intercal\Sigma^{-1}}{(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^\intercal\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)}$ [70], and show that this is equivalent to the expression for linear Fisher information in equation 4.15. The means of the response distributions along this dimension, denoted by $\hat{\mu}_i$, are $\hat{\mu}_i = \mathbf{w}^\intercal\boldsymbol{\mu}_i$ and the variance along this dimension (which we again assume is the same for both response distributions), denoted by $\hat{\sigma}_i^2$, is

$$\begin{aligned}
\hat{\sigma}_i^2 &= \text{Var}(\mathbf{w}^\intercal\mathbf{r}|s) && (4.16)\\
&= \mathbf{w}^\intercal\text{Var}(\mathbf{r}|s)\mathbf{w} && (4.17)\\
&= \mathbf{w}^\intercal\Sigma\mathbf{w} && (4.18)\\
&= \frac{(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^\intercal\Sigma^{-1}\Sigma\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)}{\left[(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^\intercal\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)\right]^2} && (4.19)\\
&= \frac{1}{(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^\intercal\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)} && (4.20)
\end{aligned}$$

and thus

$$\begin{aligned}
d'^2_{\text{SNR}} &= \frac{(\hat{\mu}_1-\hat{\mu}_0)^2}{\hat{\sigma}^2} && (4.21)\\
&= \left[\mathbf{w}^\intercal(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)\right]^2\left[(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^\intercal\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)\right] && (4.22)\\
&= (\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^\intercal\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0) && (4.23)\\
&= I && (4.24)
\end{aligned}$$

Estimating information saturation. The maximum amount of information contained in an infinitely large neural population was estimated by fitting the following saturating function to the information scaling curves in figure 4.3:

$$d'^2 = \frac{bS}{a+S} \tag{4.25}$$

where S is the ensemble size, a is the saturation rate and b is the asymptotic information value (table 4.1). The parameters a and b were fit using the *fminsearch* function in MATLAB.

4.4.3 Simulated data generation

We first evaluated decoding algorithm performance by simulating the responses of N neurons over T trials, where the population response $\mathbf{r}_i$ on trial i was generated as a sum of five terms: 1) a bias; 2) the stimulus $s_i \in \{\pm 1\}$, coupled to the population via $\boldsymbol{\alpha}$; 3) a collection of K latent variables $z_i^k \sim \mathcal{N}(0,1)$ coupled to the population via $\boldsymbol{\beta}$; 4) a $(K+1)^{\text{st}}$ latent variable $z_i^{(K+1)} \sim \mathcal{N}(0,1)$ that points in the coding direction $\boldsymbol{\alpha}$ with strength d, to explicitly introduce information-limiting noise correlations [93]; and 5) and a noise term ϵ_i:

$$\mathbf{r}_i = c\mathbf{1}_N + s_i\boldsymbol{\alpha} + \sum_{k=1}^{K} z_i^k \boldsymbol{\beta}_k + dz_i^{(K+1)}\boldsymbol{\alpha} + \boldsymbol{\epsilon}_i \tag{4.26}$$

where $\mathbf{1}_N$ is a vector of N 1s. We assume that all statistical quantities in equation 4.26 are independent of each other. Data generated in this way results in a single noise covariance matrix that is independent of the stimulus identity:

$$\text{Cov}(\mathbf{r}_i|s_i) = \sum_{k=1}^{K} \boldsymbol{\beta}_k\boldsymbol{\beta}_k^\intercal + d^2\boldsymbol{\alpha}\boldsymbol{\alpha}^\intercal + \sigma_\epsilon^2 I \tag{4.27}$$

In this setting, linear discriminant analysis is equivalent to the optimal decoder, and the population response only contains linear information [92]. We introduced nonlinear information into the population via stimulus-dependent noise covariance matrices, which requires a separate, independent set of latent variable coupling

Figure	c	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}_k$	d	$\boldsymbol{\epsilon}_i$	Stim-dep	Rect/Poiss
4.2A-C 4.8A	0	$\mathcal{N}(0, 0.25)$	$\mathcal{N}(0, 0.5)$	0.07	$\mathcal{N}(0, 1)$	No	No
4.2D-F 4.8B	0	$\mathcal{N}(0, 0.25)$	$\mathcal{N}(0, 0.5)$	0.07	$\mathcal{N}(0, 1)$	Yes	No
4.2G-I 4.8C	1	$\mathcal{N}(0, 0.0056)$	$\mathcal{N}(0, 0.5)$	0.07	$\mathcal{N}(0, 0.01)$	Yes	Yes

Table 4.2: **Simulated data details.** The performance of various decoders evaluated on these simulated datasets is shown in figures 4.2 and 4.8. All datasets were generated using $N = 200$ neurons, $K = 10$ latent variables and $T = 100000$ trials.

vectors $\{\boldsymbol{\beta}_k^j\}$ for each stimulus value j, so that

$$\text{Cov}(\mathbf{r}_i | s_i = j) = \sum_{k=1}^{K} (\boldsymbol{\beta}_k^j)(\boldsymbol{\beta}_k^j)^\intercal + d^2 \boldsymbol{\alpha}\boldsymbol{\alpha}^\intercal + \sigma_\epsilon^2 I \tag{4.28}$$

To generate data more closely resembling neural activity, for some analyses we rectified the values of $\mathbf{r}_i$, and the resulting non-negative values were used as rate parameters for independent Poisson processes to produce spiking activity. Details of each simulation are shown in table 4.2.

4.4.4 Experimental data

PFC dataset. Two macaque monkeys were trained to fixate on a central dot for 400-800 ms, after which a saccade target appeared either to the right or left of the fixation point. The monkey then had to saccade to the target and hold fixation for 500 ms before receiving a stochastic reward on correct trials. Recordings were performed with eight bilaterally implanted Utah arrays (four per hemisphere) in the prefrontal cortex (area 46). For more information see [205]. Unless otherwise noted, analysis was performed on spike counts in the 500 ms window following fixation on

the saccade target. Spike counts were z-scored in 200-trial blocks to correct for any long-term drift in firing rates. Units that had zero spikes during any 200-trial block were excluded from further analysis.

V1 dataset. The latent variable decoder was also evaluated on data from the Kohn lab, which have been made publicly available at `http://dx.doi.org/10.6080/K0NC5Z4X`. Spiking activity was recorded with a Utah array in primary visual cortex from three anesthetized macaques in response to full-contrast drifting gratings with 12 equally-spaced orientations (200 repeats). Each grating was presented for 1280 ms, and analysis was performed on spike counts between 500 ms and 1000 ms after stimulus onset to avoid transient dynamics in pairwise correlations [191]. Decoding analyses were performed on adjacent pairs of grating orientations (e.g. 0° versus 30°) for a total of 12 pairs per monkey. Spike counts were z-scored across each pair of grating orientations before decoding.

Subsampling trials and neurons. A main goal of this study was to understand how the performance of different decoders scaled with the number of neurons and the number of trials. When subsampling neurons, a random subset was chosen without replacement. When subsampling trials, a random subset was chosen without replacement to match the proportion of left vs right trials in the full dataset (a 50/50 split for both PFC and V1 datasets). For experimental data, the number of times these subsampling procedures were independently repeated is noted in the figure legends. For simulated data, subsampling was performed once for each of 25 different datasets. Training, testing and cross-validation indices were randomly assigned for each subsampling of neurons and trials.

Shuffling responses across trials. To understand the effects of noise correlations on information encoding, trials were shuffled to destroy the noise correlation structure. For each unit, spike counts were randomly permuted within each stimulus condition, and permutations were independent across units and conditions. This left each unit's first-order statistics constant, but removed any second-order statistics between pairs of units. This trial shuffling procedure was performed after each subsampling of trials. Though first shuffling the data across all possible trials would more completely remove noise correlations, shuffling the data after subsampling the trials better reflects decoding results that could be expected if only the subsampled trials were actually recorded.

4.5 Appendix: Single latent variable example

This section provides a deeper analysis of the single latent variable example introduced in sections 4.2.1 and 4.2.2. To restate the problem formulation, the population firing rate vector $\mathbf{r}_i \in \mathbb{R}^N$ on trial i is the sum of three terms: 1) the stimulus $s_i \in \{\pm 1\}$, coupled to the population via $\boldsymbol{\alpha}$; 2) a latent variable $z_i \sim \mathcal{N}(0, \sigma_z^2)$ coupled to the population via $\boldsymbol{\beta}$; 3) and a noise term $\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \sigma_\epsilon^2 I)$:

$$\mathbf{r}_i = s_i \boldsymbol{\alpha} + z_i \boldsymbol{\beta} + \boldsymbol{\epsilon}_i \tag{4.29}$$

We assume that z_i and $\boldsymbol{\epsilon}_i$ are independent, so that $\mathrm{Cov}(z_i, \boldsymbol{\epsilon}_i) = 0$. To facilitate the derivations below, we make the further assumptions that $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are unit vectors (more generally, the magnitude of each vector can be absorbed into the scalars s_i and z_i), and that $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are known. The covariance matrix of population activity, conditioned on s_i, is given by

$$\mathrm{Cov}(\mathbf{r}_i | s_i) = \Sigma = \sigma_z^2 \boldsymbol{\beta}\boldsymbol{\beta}^\intercal + \sigma_\epsilon^2 I \tag{4.30}$$

In the remainder of this section we derive an analytic "latent variable" estimator for s_i under these specific assumptions and examine its statistical properties in relation to the optimal linear estimator.

A latent variable estimator for s_i. The optimal linear estimator for s_i, denoted by $\hat{s}_i^{\mathrm{OLE}}$, is

$$\hat{s}_i^{\mathrm{OLE}} = s_{\mathrm{o}} + \frac{\boldsymbol{\alpha}\Sigma^{-1}}{\boldsymbol{\alpha}^\intercal \Sigma^{-1} \boldsymbol{\alpha}} (\mathbf{r}_i - \bar{r}) \tag{4.31}$$

where s_{o} is the average stimulus value (0 in this case) and $\bar{r} = \frac{1}{T}\sum_{i=1}^{T} \mathbf{r}_i$ [70]. We propose to exploit our knowledge of the structure of Σ in equation 4.30 to derive

a different estimator for s_i. We will first infer the activity of the latent variable z_i in the direction of $\boldsymbol{\alpha}$, then remove this component from $\mathbf{r}_i$ before decoding in the direction of $\boldsymbol{\alpha}$.

We can infer the latent variable z_i by projecting the response vector onto $\boldsymbol{\alpha}_\perp$, the component of $\boldsymbol{\beta}$ that is orthogonal to $\boldsymbol{\alpha}$:

$$\boldsymbol{\alpha}_\perp \equiv \boldsymbol{\beta} - (\boldsymbol{\alpha}^\intercal \boldsymbol{\beta})\boldsymbol{\alpha} \equiv \boldsymbol{\beta} - \gamma\boldsymbol{\alpha} \tag{4.32}$$

so that γ corresponds to the cosine of the angle between $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Then the projection of the response vector along $\boldsymbol{\alpha}_\perp$ becomes

$$\begin{aligned}
\boldsymbol{\alpha}_\perp^\intercal \mathbf{r}_i &= s_i \boldsymbol{\alpha}_\perp^\intercal \boldsymbol{\alpha} + z_i \boldsymbol{\alpha}_\perp^\intercal \boldsymbol{\beta} + \boldsymbol{\alpha}_\perp^\intercal \boldsymbol{\epsilon}_i && (4.33)\\
&= z_i \boldsymbol{\alpha}_\perp^\intercal \boldsymbol{\beta} + \boldsymbol{\alpha}_\perp^\intercal \boldsymbol{\epsilon}_i && (4.34)\\
&= z_i [\boldsymbol{\beta} - \gamma\boldsymbol{\alpha}]^\intercal \boldsymbol{\beta} + \boldsymbol{\alpha}_\perp^\intercal \boldsymbol{\epsilon}_i && (4.35)\\
&= z_i \left[1 - \gamma^2\right] + \boldsymbol{\alpha}_\perp^\intercal \boldsymbol{\epsilon}_i && (4.36)
\end{aligned}$$

Rearranging,

$$z_i = \frac{\boldsymbol{\alpha}_\perp^\intercal \mathbf{r}_i}{1-\gamma^2} - \frac{\boldsymbol{\alpha}_\perp^\intercal \boldsymbol{\epsilon}_i}{1-\gamma^2} \tag{4.37}$$

so that

$$\hat{z}_i = \frac{\boldsymbol{\alpha}_\perp^\intercal \mathbf{r}_i}{1-\gamma^2} \tag{4.38}$$

is an unbiased estimator for z_i, and $\gamma\hat{z}_i$ is an unbiased estimator for the projection of the latent variable term $z_i\boldsymbol{\beta}$ along the $\boldsymbol{\alpha}$ direction.

To arrive at the latent-variable-adjusted estimate of the stimulus, $\hat{s}_{\mathrm{LVE}}$, we simply project the population activity along the direction of $\boldsymbol{\alpha}$ and subtract the

estimate of the latent variable term in that direction:

$$\hat{s}_i^{\text{LVE}} = \boldsymbol{\alpha}^\intercal \mathbf{r}_i - \gamma \hat{z}_i \tag{4.39}$$

$$= \boldsymbol{\alpha}^\intercal \mathbf{r}_i - \gamma \frac{\boldsymbol{\alpha}_\perp^\intercal \mathbf{r}_i}{1-\gamma^2} \tag{4.40}$$

$$= \frac{[\boldsymbol{\alpha} - \gamma\boldsymbol{\beta}]^\intercal \mathbf{r}_i}{1-\gamma^2} \tag{4.41}$$

This estimate of the stimulus depends on the angle between $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ through γ, and it is instructive to note the two extreme cases. First, when $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are parallel, $\gamma = 1$ and there is no solution, because z_i cannot be disambiguated from the stimulus. Second, when $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are orthogonal, $\gamma = 0$ and the latent variable is not detrimental to decoding along $\boldsymbol{\alpha}$, so that the estimate of the stimulus reduces to $\hat{s}_i = \boldsymbol{\alpha}^\intercal \mathbf{r}_i$.

Linear Fisher information for $\hat{s}_i^{\text{LVE}}$. Given the estimate $\hat{s}_i^{\text{LVE}}$ in equation 4.41, we can calculate its linear Fisher information as the inverse of the variance of $\hat{s}_i^{\text{LVE}}$, which is given by

$$I_{LVE} = \frac{1-\gamma^2}{\sigma_\epsilon^2} \tag{4.42}$$

How does this compare to the linear Fisher information of the optimal linear estimator $\hat{s}_i^{\text{OLE}}$? By substituting equation 4.30 into the standard result that $I_{\text{OLE}} = \boldsymbol{\alpha}^\intercal \Sigma^{-1} \boldsymbol{\alpha}$,

$$I_{OLE} = \frac{1 - \frac{\gamma^2}{1+\sigma_e^2/\sigma_z^2}}{\sigma_e^2} \tag{4.43}$$

Again, we note the two extreme cases. When $\gamma = 1$, $I_{\text{LVE}} = 0$ because the latent variable is pointing in the direction of the stimulus, but I_{OLE} is greater than zero. This illustrates an important case in which $\hat{s}_i^{\text{LVE}}$ is far from optimal. When $\gamma = 0$,

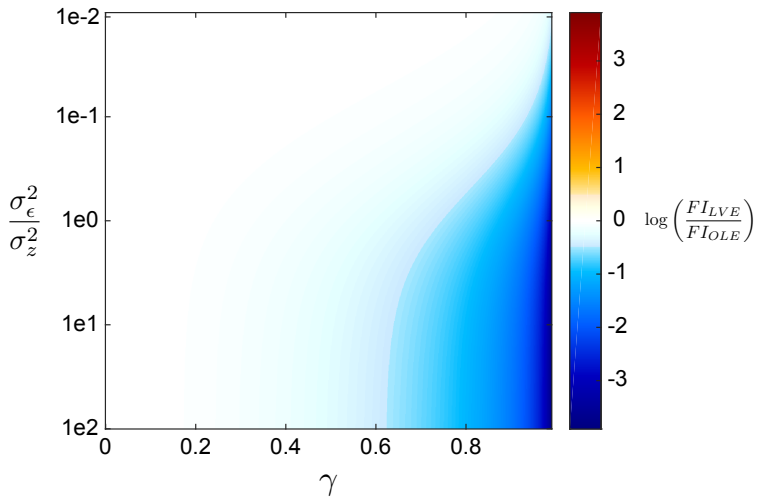

Figure 4.7: **Comparison of estimators for the single latent variable model.** Color indicates the logarithm of the ratio of the linear Fisher information for the latent variable estimator (I_{LVE}, equation 4.42) and the optimal linear estimator (I_{OLE}, equation 4.43). This value is plotted as a function of the ratio of the variances of the noise (σ_ϵ) and the latent variable (σ_z), and γ, the cosine of the angle between $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

however, I_{LVE} and I_{OLE} are equivalent. Results from intermediate values of γ are shown in figure 4.7.

Why does $I_{\text{LVE}} \to 0$ as $\gamma \to 1$? This behavior is easier to understand by considering the variance of the estimate $\hat{z}_i$, which is given by

$$\text{Var}(\hat{z}_i) = \sigma_z^2 + \frac{\sigma_\epsilon^2}{1 - \gamma^2} \tag{4.44}$$

The variance of $\hat{z}_i$ is equal to the variance of z plus a term that depends on γ. When $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are orthogonal ($\gamma = 0$), this second term becomes equal to σ_ϵ^2, the variance of the noise. As $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ become more aligned ($\gamma \to 1$), the variance of $\hat{z}_i$ blows up and drives I_{LVE} to zero.

The latent variable decoder cannot, by definition, extract more information from population responses than the optimal linear decoder. However, this single

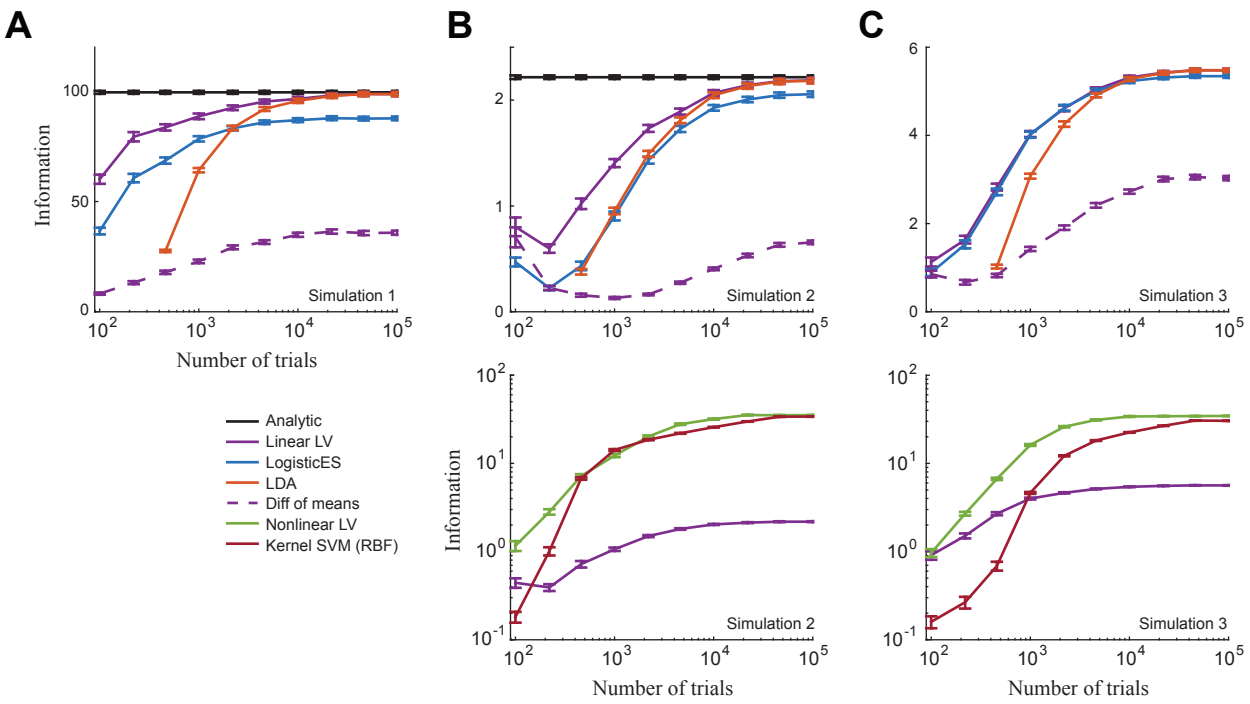

Figure 4.8: **Comparison of decoders on simulated data.** Details of the simulations are presented in section 4.4.3. **A**: Gaussian data with a single noise covariance matrix for both classes. **B**: Gaussian data with a different noise covariance matrix for each class; *top*: comparison of linear decoders; *bottom*: comparison of nonlinear decoders. **C**: Same as *B*, except resulting values are rectified and then passed through a Poisson spike generator to simulate spike count data.

latent variable example demonstrates that there are a wide range of parameter settings for which the latent variable decoder performs close to optimal. Importantly, this analysis only considers the behavior of these estimators in the limit of infinite data, and does not consider how efficiently these estimators use finite amounts of data. In practice (i.e. with a limited number of trials), the latent variable decoder is able to more efficiently extract information than a range of other decoders (see section 4.4.2) using both simulated (figure 4.8) and experimental data (figures 4.3 and 4.5).

Extension to multiple latent variables. In practice, we must estimate not

only $\hat{z}_i$, but the quantities $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as well. Section 4.4.1 explicitly describes the steps for training the decoder, but here we give a more intuitive picture of how the decoder weights might be learned.

Equation 4.39 shows that once we project the response vector $\mathbf{r}_i$ down onto the stimulus coding direction $\boldsymbol{\alpha}$ - which can be estimated from the mean responses to the different stimuli - we then subtract off an estimate of the latent variable activity in that direction. In this simple example that estimate is just a linear projection of $\mathbf{r}_i$:

$$\gamma \hat{z}_i = \frac{\gamma \boldsymbol{\alpha}_\perp^\intercal}{1-\gamma^2} \mathbf{r}_i \tag{4.45}$$

and the decoder weights $\gamma \boldsymbol{\alpha}_\perp^\intercal / (1-\gamma^2)$ can be learned by regressing $\mathbf{r}_i$ on the stimulus-subtracted residual activity $s_i - \boldsymbol{\alpha}^\intercal \mathbf{r}_i$, where s_i is the presented stimulus rather than the estimate.

The single latent variable example is now easy to extend to K latent variables, denoted by $\{z_i^k\}_{k=1}^K$. Following the same logic as above, in which we subtract out the contribution of each latent variable,

$$\begin{aligned} \hat{s}_i^{\text{LVE}} &= \boldsymbol{\alpha}^\intercal \mathbf{r}_i - \sum_{k=1}^K \gamma_k z_i^k & (4.46) \\ &= \boldsymbol{\alpha}^\intercal \mathbf{r}_i - \left[\sum_{k=1}^K \frac{\gamma_k (\boldsymbol{\alpha}_k)_\perp^\intercal}{1-\gamma_k^2} \right] \mathbf{r}_i & (4.47) \end{aligned}$$

which is again a linear projection of $\mathbf{r}_i$ that can be learned using linear regression. Note that we never explicitly estimate any of the γ_k's or $(\boldsymbol{\alpha}_k)_\perp$'s, but instead estimate their combined effect in the direction of $\boldsymbol{\alpha}$, represented by the sum in equation 4.47. It is this property of the estimator $\hat{s}_i^{\text{LVE}}$ that allows us to learn the proper projection without first specifying the number of latent variables K in the algorithm.

Chapter 5: Conclusions

5.1 Latent variable models for neuroscience

The increasing popularity of statistical models in neuroscience is driven by experimental advances that can now provide datasets of unprecedencted size and complexity. Unlike more theoretically mature fields of study such as physics, there are no solid conceptual frameworks in which to understand these data. Data analysis tools have become increasingly necessary to explore these datasets in search of meaningful structure that can guide our understanding of the principles that underlie brain structure and function, and suggest additional experimental investigations.

Latent variable models in particular have proven to be particularly effective at discovering a parsimonious description of these data across a range of species, brain regions and experimental paradigms, in both structural [206] and functional [160] data. These low-dimensional descriptions are useful for understanding the data, but also hint at the possibility that low-dimensional activity is a hallmark of neural systems in particular regimes. Indeed, the brain has evolved to process large amounts of incoming sensory information, and the intuitive idea that the brain's solutions involve some form of dimensionality reduction is attractive, though still far from understood.

The power of latent variable models (including those developed in this dissertation) are ultimately constrained by the richness of the data they attempt to explain. In this sense, despite the impressive experimental advances of the last decade, there are still a wealth of experimental limitations that remain to be overcome. Most experiments still employ extremely simple stimuli and behavioral paradigms (if any at all), which are chosen to isolate a particular mechanism or computation. This eases the resulting analysis and interpretation, but also imposes artificial constraints on a system that evolved to perform much more difficult tasks. In the future, it will be critical to ensure that stimuli and behavioral tasks are complicated enough to engage the relevant neural computations [173,194]. Another experimental limitation is the size of the neural populations being recorded. Though this number is increasing exponentially, experimentalists still only typically record from an infinitesimally small fraction of the available neurons in a single brain region (though see [117]). One question that is not well understood for most model systems is the relevant population size required to perform a given computation, and how that population is distributed throughout different parts of the brain.

As experimentalists overcome these challenges, the extreme flexibility of latent variable models will allow them to adapt to changing analysis demands. One interesting direction for future model development is incorporating data obtained across multiple spatial and temporal scales, such as simultaneous wide-field imaging of large brain regions and more focal electrophysiology recordings, to understand the flow of information through neural circuits. Of course, this also raises the question of how to determine the coarsest level of spatial and temporal resolution needed to

gain fundamental insights into a particular computation. Another interesting direction is incorporating connectomics data that describes the structural connections between different neurons or brain regions. This data will allow for the combination of latent variable models with information about the strucutre of the circuit to further constrain the space in which we seek solutions, to arrive at a more holistic understanding of how structure gives rise to function.

5.2 The role of feedback in perception

The latent variable models developed in this dissertation explicitly seek to explain trial-to-trial variability in sensory neural responses. This approach is in contrast to classical models of sensory processing, which are feed-forward in nature and thus (by definition) learn a deterministic mapping from stimulus to response. Models that are able to capture trial-to-trial variability will be important tools for understanding the mechanisms that give rise to perception, one of the most important functions of any nervous system.

Sensation refers to the process of gathering information about the external environment through a variety of sensors. Rods and cones respond to patterns of light falling on the retina, and give rise to visual sensation. Other receptors collect signals from different sensory domains, including sound, touch, taste, and smell. *Perception* is the interpretation of these signals by the brain, and allows sensory signals to be processed in a context-dependent manner.

One particularly striking example of the distinction between sensation and

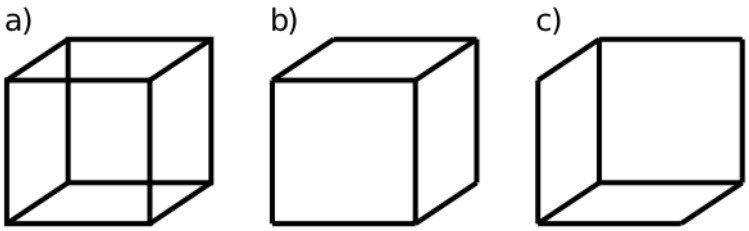

Figure 5.1: **Necker cube illusion.** The Necker Cube (**A**) is a bistable percept that appears as either (**B**) or (**C**) at any given point in time without a change in the visual input. Image from `https://grey.colorado.edu/CompCogNeuro/index.php/CCNBook/Sims/Networks/Necker_Cube`.

perception is given by the Necker Cube illusion (figure 5.1); this illusion is known as a *bistable percept*, in which one of two possible percepts is experienced at any given time, but the two can never be fused into a single stable percept. Regardless of which percept is experienced at any given time, the sensory information is exactly identical.

If the sensory information is identical but the perception of the cube is constantly changing, what drives the differing interpretations? One well-supported hypothesis is that perception is driven by internally-generated *feedback* signals from different brain regions. Anatomical studies have shown there is an immense amount of lateral and feedback projections within the visual system [207], which provide the anatomical substrate by which feedback signals can be introduced into sensory cortex. Additional studies have revealed the presence of functional connectivity between non-sensory areas like prefrontal cortex and the earliest stages of cortical visual processing [208, 209], though it is still unclear if this functional connectivity is supported by direct structural connectivity. This feedback activity, both direct and indirect, results in variability in sensory neural responses, and may give rise to

the transformation from sensation to perception.

In addition to the anatomical evidence, several other lines of research implicate feedback activity in perception by studying scenarios where mental imagery is generated without external stimulation (i.e. perception without sensation). For example, a recent study on mental imagery demonstrated high similarity of neural representations in higher-level visual areas between imagined and stimulus-driven percepts [210]. Another recent study demonstrated increased feedback activity from fronto-parietal to visual areas during both perception and mental imagery, and this feedback activity was both stronger during mental imagery and increased in early visual areas with an increase in self-reported vividness of the imagery [209], exposing a possible top-down mechanism that differentiates perception and mental imagery. A study similar to [210] has shown a similarity of neural representations between dreaming and perception [211], and other studies have sought to understand whether dreaming is closer to perception or mental imagery [212]. Additonal work on the mechanisms that underlie visual hallucinations also strongly implicate feedback projections in this perceptual phenomenon [213, 214].

The previous paragraphs illustrate several situations in which feedback is implicated in perception, but what role does feedback actually play? One theoretical framework that has received significant attention over the previous few decades assumes the brain constructs a generative model of the environment (an *internal model*) and uses this model to infer the underlying causes of the incoming sensory signals, which can then be used to inform decision-making processes and behavior.

The theory of predictive coding [215] is one such instantiation of this frame-

work, and uses feedback to represent a prediction about the state of the external environment, generated through the internal model. This prediction is then compared to the incoming sensory information, and the mismatch between the two, called the prediction error, is used to update the internal model. In this framework, then, feed-forward activity represents the prediction error rather than the stimulus itself. A model of predictive coding in the visual system led to the development of ubiquitous receptive field structures, and could also explain several empirical observations of receptive fields such as end-stopping [216].

Bayesian inference is another instantiation of this framework that is related to predictive coding [217, 218]. In this case feedback (the prior) still represents a prediction about the external world, but this prediction is combined with the observations (the likelihood) to update the internal model (posterior) through approximate Bayesian inference.

In a sense Bayesian inference and predictive coding are complimentary ideas about how internal models of the environment can be combined with sensory information; predictive coding posits that feed-forward neural activity should represent prediction errors rather than the the prediction itself, and does not address how that prediction is made or how prediction errors are utilized. Bayesian inference, on the other hand, specifies how predictions should be computed and updated with incoming sensory information (Bayes rule) without specifying how these computations are performed in neural populations (though evidence from multi-sensory integration [219] and sensorimotor learning [220] point to this capacity). However, recent work has sought to bring these two ideas together into a "Bayesian predictive

coding" framework [221].

Understanding the role of feedback in perception will require the continued development of these theoretical models, guided by the analysis of experimental data. These analyses, in turn, will require sophisticated models of sensory neurons that are capable of capturing the variability that ostensibly underlies perception.

5.3 Understanding the brain

As the previous sections have argued, latent variable models that are able to describe variability in neural responses will be crucial for understanding perception, as well as various other neural processes. Here, I want to take a step back and consider what it means to "understand" a system. One perspective was suggested by the influential neuroscientist David Marr, who proposed that the brain (and the visual system in particular) should be studied in the context of the problems it needs to solve. For this reason he viewed the brain abstractly as an information processing system, which could be understood at three complementary levels of analysis [55]:

- **computational**: at this level, the questions of interest are *what* is the problem that needs to be solved and *why* does it need to be solved?
- **algorithmic**: what are the representations of the input and output information used to solve the problem, and how are those representations manipulated by the system?
- **implementation**: what is the physical substrate that supports those representations?

Marr's main intuition was that, although "algorithms and mechanisms are empirically more accessible,...the level of computational theory...is critically important from an information-processing point of view...[because]...the nature of the computations that underlie perception depends more upon the computational problems that have to be solved than upon the particular hardware in which their solutions are implemented" [55]. Despite more than three decades of research since Marr's statement, progress in computational theories of brain function continue to lag far behind experimental capabilities, which typically constitute understanding at the implementation level. This is true even of a highly-studied system like vision; Bruno Olshausen, in a review entitled "20 years of learning about vision" states "the problem is not just that we lack the proper data, but that we do not even have the right conceptual framework for thinking about what is happening" [222].

In this dissertation I have argued for the usefulness of statistical models in exploring neural data, with the ultimate goal of discovering fundamental principles of information processing. However, a more modest and realistic goal is closer to "suggesting" fundamental principles rather than "discovering", which will require data analysis, theory, and experiment working in concert to inform each other. Indeed, even sophisticated data-driven approaches cannot always be successful in suggesting useful research directions. In a controversial study entitled "Could a neuroscientist understand a microprocessor?", Jonas and Kording [223] utilized a relatively simple microprocessor as a model organism to test an array of analysis tools currently used in neuroscience. Though they were able to discover "interesting" structure in the data, their analyses ultimately failed to uncover the information

processing strategies used by the microprocessor. This study cautions that even if we have infinite amounts of data from all relevant components of a relatively simple model organism, if our analyses are not well constrained by computational theories of information processing, we will be swimming in information without a means to extract the relevant knowledge.

Despite this cautionary tale and others [224], the field of neuroscience is aware of these shortcomings and taking steps to fix them [225, 226]. Significant progress in our understanding of brain structure and function will require interdisciplinary collaborations and interdisciplinary training for young neuroscientists. A holistic approach to studying the brain should integrate perspectives and tools from fields that traditionally study the brain, such as systems neuroscience, cognitive neuroscience, psychology and the philosophy of mind, as well as more quantitative fields such as electrical engineering, statistics, machine learning, and artificial intelligence. Only with these combined perspectives, across each of Marr's levels of analysis, will we begin to unlock the mysteries of the mind.

Bibliography

[1] Terrence J Sejnowski, Patricia S Churchland, and J Anthony Movshon. Putting big data to good use in neuroscience. *Nature neuroscience*, 17(11):1440–1441, 2014.

[2] David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591, 1959.

[3] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *nature*, 503(7474):78, 2013.

[4] Emery N Brown, Patrick L Purdon, and Christa J Van Dort. General anesthesia and altered states of arousal: a systems neuroscience analysis. *Annual review of neuroscience*, 34:601–628, 2011.

[5] Nicole C Rust and J Anthony Movshon. In praise of artifice. *Nature neuroscience*, 8(12):1647, 2005.

[6] Niru Maheswaranathan, Lane McIntosh, David Kastner, Luke Brezovec, Aran Nayebi, Surya Ganguli, and Stephen Baccus. Deep models of retinal responses to natural scenes generalize to diverse structured stimuli. In *Computational and Systems Neuroscience*, 2018.

[7] Ian H Stevenson and Konrad P Kording. How advances in neural recording affect data analysis. *Nature neuroscience*, 14(2):139, 2011.

[8] Peter Dayan and Laurence F Abbott. *Theoretical neuroscience*, volume 806. Cambridge, MA: MIT Press, 2001.

[9] Mary M Heinricher. Principles of extracellular single-unit recording.

[10] James J Jun, Nicholas A Steinmetz, Joshua H Siegle, Daniel J Denman, Marius Bauza, Brian Barbarits, Albert K Lee, Costas A Anastassiou, Alexandru Andrei, Çağatay Aydın, et al. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551(7679):232, 2017.

[11] Joshua T Vogelstein, Brendon O Watson, Adam M Packer, Rafael Yuste, Bruno Jedynak, and Liam Paninski. Spike inference from calcium imaging using sequential monte carlo methods. *Biophysical journal*, 97(2):636–655, 2009.

[12] Joshua T Vogelstein, Adam M Packer, Timothy A Machado, Tanya Sippy, Baktash Babadi, Rafael Yuste, and Liam Paninski. Fast nonnegative deconvolution for spike train inference from population calcium imaging. *Journal of neurophysiology*, 104(6):3691–3704, 2010.

[13] Marius Pachitariu, Carsen Stringer, Sylvia Schröder, Mario Dipoppa, L Federico Rossi, Matteo Carandini, and Kenneth D Harris. Suite2p: beyond 10,000 neurons with standard two-photon microscopy. *Biorxiv*, page 061507, 2016.

[14] Andrea Giovannucci, Johannes Friedrich, Matt Kaufman, Anne Churchland, Dmitri Chklovskii, Liam Paninski, and Eftychios A Pnevmatikakis. Onacid: Online analysis of calcium imaging data in real time. In *Advances in Neural Information Processing Systems*, pages 2378–2388, 2017.

[15] Lucas Theis, Philipp Berens, Emmanouil Froudarakis, Jacob Reimer, Miroslav Román Rosón, Tom Baden, Thomas Euler, Andreas S Tolias, and Matthias Bethge. Benchmarking spike rate inference in population calcium imaging. *Neuron*, 90(3):471–482, 2016.

[16] Valentina Emiliani, Adam E Cohen, Karl Deisseroth, and Michael Häusser. All-optical interrogation of neural circuits. *Journal of Neuroscience*, 35(41):13917–13926, 2015.

[17] CM Bishop. Pattern recognition and machine learning: springer new york. 2006.

[18] Liam Paninski. Maximum likelihood estimation of cascade point-process neural encoding models. *Network: Computation in Neural Systems*, 15(4):243–262, 2004.

[19] Jonathan W Pillow, Jonathon Shlens, Liam Paninski, Alexander Sher, Alan M Litke, EJ Chichilnisky, and Eero P Simoncelli. Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, 454(7207):995, 2008.

[20] James M McFarland, Yuwei Cui, and Daniel A Butts. Inferring nonlinear neuronal computation based on physiologically plausible inputs. *PLoS Comput Biol*, 9(7):e1003143, 2013.

[21] Ross S Williamson, Maneesh Sahani, and Jonathan W Pillow. The equivalence of information-theoretic and likelihood-based methods for neural dimensionality reduction. *PLoS computational biology*, 11(4):e1004141, 2015.

[22] James Scott and Jonathan W Pillow. Fully bayesian inference for neural models with negative-binomial spiking. In *Advances in neural information processing systems*, pages 1898–1906, 2012.

[23] Yuanjun Gao, Evan W Archer, Liam Paninski, and John P Cunningham. Linear dynamical neural population models through nonlinear embeddings. In *Advances in Neural Information Processing Systems*, pages 163–171, 2016.

[24] Matthew R Whiteway and Daniel A Butts. Revealing unobserved factors underlying cortical activity with a rectified latent variable model applied to neural population recordings. *Journal of neurophysiology*, 117(3):919–936, 2017.

[25] Peter McCullagh and John A Nelder. *Generalized linear models*. Springer, 1989.

[26] Yuwei Cui, Liu D Liu, James M McFarland, Christopher C Pack, and Daniel A Butts. Inferring cortical variability from local field potentials. *Journal of Neuroscience*, 36(14):4121–4135, 2016.

[27] Leigh R Hochberg, Daniel Bacher, Beata Jarosiewicz, Nicolas Y Masse, John D Simeral, Joern Vogel, Sami Haddadin, Jie Liu, Sydney S Cash, Patrick van der Smagt, et al. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398):372, 2012.

[28] Daniel L. K. Yamins and James J. DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, 19:356–365, 2016.

[29] Christopher R Holdgraf, Jochem W Rieger, Cristiano Micheli, Stephanie Martin, Robert T Knight, and Frederic E Theunissen. Encoding and decoding models in cognitive electrophysiology. *Frontiers in systems neuroscience*, 11:61, 2017.

[30] Stephen V David, William E Vinje, and Jack L Gallant. Natural stimulus statistics alter the receptive field structure of v1 neurons. *Journal of Neuroscience*, 24(31):6991–7006, 2004.

[31] Bruno A Olshausen and David J Field. Sparse coding of sensory inputs. *Current opinion in neurobiology*, 14(4):481–487, 2004.

[32] Jean Pierre Richard, Hans-Joachim Leppelsack, and Martine Hausberger. A rapid correlation method for the analysis of spectro-temporal receptive fields of auditory neurons. *Journal of neuroscience methods*, 61(1-2):99–103, 1995.

[33] Jan Kubanek, Peter Brunner, Aysegul Gunduz, David Poeppel, and Gerwin Schalk. The tracking of speech envelope in the human cortex. *PloS one*, 8(1):e53398, 2013.

[34] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[35] Edward H Adelson and James R Bergen. Spatiotemporal energy models for the perception of motion. *Josa a*, 2(2):284–299, 1985.

[36] Matteo Carandini, Jonathan B Demb, Valerio Mante, David J Tolhurst, Yang Dan, Bruno A Olshausen, Jack L Gallant, and Nicole C Rust. Do we know what the early visual system does? *Journal of Neuroscience*, 25(46):10577–10597, 2005.

[37] Misha B Ahrens, Liam Paninski, and Maneesh Sahani. Inferring input nonlinearities in neural encoding models. *Network: Computation in Neural Systems*, 19(1):35–67, 2008.

[38] Trevor Hastie and Robert Tibshirani. *Generalized additive models*. Wiley Online Library, 1990.

[39] Il Memming Park, Evan W Archer, Nicholas Priebe, and Jonathan W Pillow. Spectral methods for neural characterization using generalized quadratic models. In *Advances in neural information processing systems*, pages 2454–2462, 2013.

[40] Kanaka Rajan, Olivier Marre, and Gašper Tkačik. Learning quadratic receptive fields from neural responses to natural stimuli. *Neural computation*, 25(7):1661–1692, 2013.

[41] Brian Lau, Garrett B Stanley, and Yang Dan. Computational subunits of visual cortical neurons revealed by artificial neural networks. *Proceedings of the National Academy of Sciences*, 99(13):8974–8979, 2002.

[42] Ryan Prenger, Michael C-K Wu, Stephen V David, and Jack L Gallant. Nonlinear v1 responses to natural scenes revealed by neural network analysis. *Neural Networks*, 17(5-6):663–679, 2004.

[43] Simon P Peron, Jeremy Freeman, Vijay Iyer, Caiying Guo, and Karel Svoboda. A cellular resolution map of barrel cortex activity during tactile behavior. *Neuron*, 86(3):783–799, 2015.

[44] Adam H Marblestone, Greg Wayne, and Konrad P Kording. Toward an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience*, 10:94, 2016.

[45] Tim Christian Kietzmann, Patrick McClure, and Nikolaus Kriegeskorte. Deep neural networks in computational neuroscience. *bioRxiv*, page 133504, 2017.

[46] Lane McIntosh, Niru Maheswaranathan, Aran Nayebi, Surya Ganguli, and Stephen Baccus. Deep learning models of the retinal response to natural scenes. In *Advances in neural information processing systems*, pages 1369–1377, 2016.

[47] Eleanor Batty, Josh Merel, Nora Brackbill, Alexander Heitman, Alexander Sher, Alan Litke, EJ Chichilnisky, and Liam Paninski. Multilayer recurrent network models of primate retinal ganglion cell responses. 2016.

[48] Ján Antolík, Sonja B Hofer, James A Bednar, and Thomas D Mrsic-Flogel. Model constrained by visual hierarchy improves prediction of neural responses to natural scenes. *PLoS computational biology*, 12(6):e1004927, 2016.

[49] David Klindt, Alexander S Ecker, Thomas Euler, and Matthias Bethge. Neural system identification for large populations separating what and where. In *Advances in Neural Information Processing Systems*, pages 3509–3519, 2017.

[50] Santiago A Cadena, George H Denfield, Edgar Y Walker, Leon A Gatys, Andreas S Tolias, Matthias Bethge, and Alexander S Ecker. Deep convolutional models improve predictions of macaque v1 responses to natural images. *bioRxiv*, page 201764, 2017.

[51] William F Kindel, Elijah D Christensen, and Joel Zylberberg. Using deep learning to reveal the neural code for images in primary visual cortex. *arXiv preprint arXiv:1706.06208*, 2017.

[52] Michael Oliver and Jack Gallant. A deep convolutional energy model of v4 responses to natural movies. *Journal of Vision*, 16(12):876–876, 2016.

[53] Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019, 1999.

[54] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[55] David Marr. Vision: A computational approach, 1982.

[56] Surya Ganguli and Haim Sompolinsky. Compressed sensing, sparsity, and dimensionality in neuronal information processing and data analysis. *Annual review of neuroscience*, 35:485–508, 2012.

[57] Sebastian Gerwinn, Jakob H Macke, and Matthias Bethge. Bayesian inference for generalized linear models for spiking neurons. *Frontiers in computational neuroscience*, 4:12, 2010.

[58] Ana Calabrese, Joseph W Schumacher, David M Schneider, Liam Paninski, and Sarah MN Woolley. A generalized linear model for estimating spectrotemporal receptive fields from responses to natural sounds. *PloS one*, 6(1):e16104, 2011.

[59] Frédéric E Theunissen, Stephen V David, Nandini C Singh, Anne Hsu, William E Vinje, and Jack L Gallant. Estimating spatio-temporal receptive fields of auditory and visual neurons from their responses to natural stimuli. *Network: Computation in Neural Systems*, 12(3):289–316, 2001.

[60] Maneesh Sahani and Jennifer F Linden. Evidence optimization techniques for estimating stimulus-response functions. In *Advances in neural information processing systems*, pages 317–324, 2003.

[61] Timm Lochmann, Timothy J Blanche, and Daniel A Butts. Construction of direction selectivity through local energy computations in primary visual cortex. *PloS one*, 8(3):e58666, 2013.

[62] Mijung Park and Jonathan W Pillow. Receptive field inference with localized priors. *PLoS computational biology*, 7(10):e1002219, 2011.

[63] Jonathan W Pillow, Liam Paninski, Valerie J Uzzell, Eero P Simoncelli, and EJ Chichilnisky. Prediction and decoding of retinal ganglion cell responses with a probabilistic spiking model. *Journal of Neuroscience*, 25(47):11003–11013, 2005.

[64] Yuwei Cui, Yanbin V Wang, Silvia JH Park, Jonathan B Demb, and Daniel A Butts. Divisive suppression explains high-precision firing and contrast adaptation in retinal ganglion cells. *eLife*, 5:e19460, 2016.

[65] Daniel A Butts, Chong Weng, Jianzhong Jin, Jose-Manuel Alonso, and Liam Paninski. Temporal precision in the visual pathway through the interplay of excitation and stimulus-driven suppression. *Journal of Neuroscience*, 31(31):11313–11327, 2011.

[66] Valerio Mante, Vincent Bonin, and Matteo Carandini. Functional mechanisms shaping lateral geniculate responses to artificial and natural stimuli. *Neuron*, 58(4):625–638, 2008.

[67] Robbe LT Goris, J Anthony Movshon, and Eero P Simoncelli. Partitioning neuronal variability. *Nature neuroscience*, 17(6):858, 2014.

[68] Paul Tiesinga, Jean-Marc Fellous, and Terrence J Sejnowski. Regulation of spike timing in visual cortical circuits. *Nature reviews neuroscience*, 9(2):97, 2008.

[69] Roger Herikstad, Jonathan Baker, Jean-Philippe Lachaux, Charles M Gray, and Shih-Cheng Yen. Natural movies evoke spike trains with low spike time variability in cat primary visual cortex. *Journal of Neuroscience*, 31(44):15844–15860, 2011.

[70] Adam Kohn, Ruben Coen-Cagli, Ingmar Kanitscheider, and Alexandre Pouget. Correlations and neuronal population information. *Annual review of neuroscience*, 39:237–256, 2016.

[71] A Aldo Faisal, Luc PJ Selen, and Daniel M Wolpert. Noise in the nervous system. *Nature reviews neuroscience*, 9(4):292, 2008.

[72] Zachary F Mainen and Terrence J Sejnowski. Reliability of spike timing in neocortical neurons. *Science*, 268(5216):1503–1506, 1995.

[73] Pierre-Simon Laplace. *Essai philosophique sur les probabilités*. H. Remy, 1829.

[74] Horace B Barlow. Single units and sensation: a neuron doctrine for perceptual psychology? *Perception*, 1(4):371–394, 1972.

[75] Marlene R Cohen and Adam Kohn. Measuring and interpreting neuronal correlations. *Nature neuroscience*, 14(7):811–819, 2011.

[76] Horace Barlow. Redundancy reduction revisited. *Network: computation in neural systems*, 12(3):241–253, 2001.

[77] Adam Kohn and Matthew A Smith. Stimulus dependence of neuronal correlation in primary visual cortex of the macaque. *Journal of Neuroscience*, 25(14):3661–3673, 2005.

[78] Adrián Ponce-Alvarez, Alexander Thiele, Thomas D Albright, Gene R Stoner, and Gustavo Deco. Stimulus-dependent variability and noise correlations in cortical mt neurons. *Proceedings of the National Academy of Sciences*, 110(32):13162–13167, 2013.

[79] Marlene R Cohen and William T Newsome. Context-dependent changes in functional circuitry in visual area mt. *Neuron*, 60(1):162–173, 2008.

[80] Adrian G Bondy, Ralf M Haefner, and Bruce G Cumming. Feedback determines the structure of correlated variability in primary visual cortex. *Nature neuroscience*, page 1, 2018.

[81] Alexander S Ecker, Philipp Berens, R James Cotton, Manivannan Subramaniyan, George H Denfield, Cathryn R Cadwell, Stelios M Smirnakis, Matthias Bethge, and Andreas S Tolias. State dependence of noise correlations in macaque primary visual cortex. *Neuron*, 82(1):235–248, 2014.

[82] Marieke L Schölvinck, Aman B Saleem, Andrea Benucci, Kenneth D Harris, and Matteo Carandini. Cortical state determines global variability and correlations in visual cortex. *Journal of Neuroscience*, 35(1):170–178, 2015.

[83] Takaki Komiyama, Takashi R Sato, Daniel H OConnor, Ying-Xin Zhang, Daniel Huber, Bryan M Hooks, Mariano Gabitto, and Karel Svoboda. Learning-related fine-scale specificity imaged in motor cortex circuits of behaving mice. *Nature*, 464(7292):1182, 2012.

[84] Yong Gu, Sheng Liu, Christopher R Fetsch, Yun Yang, Sam Fok, Adhira Sunkara, Gregory C DeAngelis, and Dora E Angelaki. Perceptual learning reduces interneuronal correlations in macaque visual cortex. *Neuron*, 71(4):750–761, 2011.

[85] Hiroshi Makino, Chi Ren, Haixin Liu, An Na Kim, Neehar Kondapaneni, Xin Liu, Duygu Kuzum, and Takaki Komiyama. Transformation of cortex-wide emergent properties during motor learning. *Neuron*, 94(4):880–890, 2017.

[86] AM Ni, DA Ruff, JJ Alberts, J Symmonds, and MR Cohen. Learning and attention reveal a general relationship between population activity and behavior. *Science*, 359(6374):463–465, 2018.

[87] Marlene R Cohen and John HR Maunsell. Attention improves performance primarily by reducing interneuronal correlations. *Nature neuroscience*, 12(12):1594, 2009.

[88] Jude F Mitchell, Kristy A Sundberg, and John H Reynolds. Spatial attention decorrelates intrinsic activity fluctuations in macaque area v4. *Neuron*, 63(6):879–888, 2009.

[89] Mehdi Adibi, James S McDonald, Colin WG Clifford, and Ehsan Arabzadeh. Adaptation improves neural coding efficiency despite increasing correlations in variability. *Journal of Neuroscience*, 33(5):2108–2120, 2013.

[90] Larry F Abbott and Peter Dayan. The effect of correlated variability on the accuracy of a population code. *Neural computation*, 11(1):91–101, 1999.

[91] Maoz Shamir and Haim Sompolinsky. Nonlinear population codes. *Neural computation*, 16(6):1105–1136, 2004.

[92] Bruno B Averbeck and Daeyeol Lee. Effects of noise correlations on information encoding and decoding. *Journal of neurophysiology*, 95(6):3633–3644, 2006.

[93] Rubén Moreno-Bote, Jeffrey Beck, Ingmar Kanitscheider, Xaq Pitkow, Peter Latham, and Alexandre Pouget. Information-limiting correlations. *Nature neuroscience*, 17(10):1410, 2014.

[94] Ingmar Kanitscheider, Ruben Coen-Cagli, and Alexandre Pouget. Origin of information-limiting noise correlations. *Proceedings of the National Academy of Sciences*, 112(50):E6973–E6982, 2015.

[95] Alexander S Ecker, George H Denfield, Matthias Bethge, and Andreas S Tolias. On the structure of neuronal population activity under fluctuations in attentional state. *Journal of Neuroscience*, 36(5):1775–1789, 2016.

[96] I-Chun Lin, Michael Okun, Matteo Carandini, and Kenneth D Harris. The nature of shared cortical variability. *Neuron*, 87(3):644–656, 2015.

[97] Marius Pachitariu, Dmitry R Lyamzin, Maneesh Sahani, and Nicholas A Lesica. State-dependent population coding in primary auditory cortex. *Journal of Neuroscience*, 35(5):2058–2073, 2015.

[98] Iñigo Arandia-Romero, Seiji Tanabe, Jan Drugowitsch, Adam Kohn, and Rubén Moreno-Bote. Multiplicative and additive modulation of neuronal tuning with population activity affects encoded information. *Neuron*, 89(6):1305–1316, 2016.

[99] Charles G Frye and Jason N MacLean. Spontaneous activations follow a common developmental course across primary sensory areas in mouse neocortex. *Journal of neurophysiology*, 116(2):431–437, 2016.

[100] Amos Arieli, Alexander Sterkin, Amiram Grinvald, and AD Aertsen. Dynamics of ongoing activity: explanation of the large variability in evoked cortical responses. *Science*, 273(5283):1868–1871, 1996.

[101] Neil C Rabinowitz, Robbe L Goris, Marlene Cohen, and Eero P Simoncelli. Attention stabilizes the shared gain of v4 populations. *Elife*, 4:e08998, 2015.

[102] Michael Okun, Nicholas A Steinmetz, Lee Cossell, M Florencia Iacaruso, Ho Ko, Péter Barthó, Tirin Moore, Sonja B Hofer, Thomas D Mrsic-Flogel, Matteo Carandini, et al. Diverse coupling of neurons to populations in sensory cortex. *Nature*, 521(7553):511, 2015.

[103] Timothée Masquelier. Neural variability, or lack thereof. *Frontiers in computational neuroscience*, 7:7, 2013.

[104] Emili Balaguer-Ballester. Cortical variability and challenges for modeling approaches. *Frontiers in systems neuroscience*, 11:15, 2017.

[105] Richard D Lange and Ralf M Haefner. Characterizing and interpreting the influence of internal variables on sensory activity. *Current opinion in neurobiology*, 46:84–89, 2017.

[106] Anne C Smith and Emery N Brown. Estimating a state-space model from point process observations. *Neural Computation*, 15(5):965–991, 2003.

[107] Liam Paninski, Yashar Ahmadian, Daniel Gil Ferreira, Shinsuke Koyama, Kamiar Rahnama Rad, Michael Vidne, Joshua Vogelstein, and Wei Wu. A new look at state-space models for neural data. *Journal of computational neuroscience*, 29(1-2):107–126, 2010.

[108] Jakob H Macke, Lars Buesing, John P Cunningham, M Yu Byron, Krishna V Shenoy, and Maneesh Sahani. Empirical models of spiking in neural populations. In *Advances in neural information processing systems*, pages 1350–1358, 2011.

[109] Marius Pachitariu, Biljana Petreska, and Maneesh Sahani. Recurrent linear models of simultaneously-recorded neural populations. In *Advances in Neural Information Processing Systems*, pages 3138–3146, 2013.

[110] Evan W Archer, Urs Koster, Jonathan W Pillow, and Jakob H Macke. Low-dimensional models of neural population activity in sensory cortical circuits. In *Advances in Neural Information Processing Systems*, pages 343–351, 2014.

[111] M Yu Byron, John P Cunningham, Gopal Santhanam, Stephen I Ryu, Krishna V Shenoy, and Maneesh Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. In *Advances in neural information processing systems*, pages 1881–1888, 2009.

[112] Yuan Zhao and Il Memming Park. Variational latent gaussian process for recovering single-trial dynamics from population spike trains. *Neural Computation*, 2017.

[113] Biljana Petreska, M Yu Byron, John P Cunningham, Gopal Santhanam, Stephen I Ryu, Krishna V Shenoy, and Maneesh Sahani. Dynamical segmentation of single trials from population neural data. In *Advances in neural information processing systems*, pages 756–764, 2011.

[114] Scott Linderman, Matthew Johnson, Andrew Miller, Ryan Adams, David Blei, and Liam Paninski. Bayesian learning and inference in recurrent switching linear dynamical systems. In *Artificial Intelligence and Statistics*, pages 914–922, 2017.

[115] David Sussillo, Rafal Jozefowicz, LF Abbott, and Chethan Pandarinath. Lfads-latent factor analysis via dynamical systems. *arXiv preprint arXiv:1608.06315*, 2016.

[116] John P Cunningham and Zoubin Ghahramani. Linear dimensionality reduction: survey, insights, and generalizations. *Journal of Machine Learning Research*, 16(1):2859–2900, 2015.

[117] Misha B Ahrens, Jennifer M Li, Michael B Orger, Drew N Robson, Alexander F Schier, Florian Engert, and Ruben Portugues. Brain-wide neuronal dynamics during motor adaptation in zebrafish. *Nature*, 485(7399):471, 2012.

[118] Benjamin R Cowley, Matthew A Smith, Adam Kohn, and M Yu Byron. Stimulus-driven population activity patterns in macaque primary visual cortex. *PLOS Computational Biology*, 12(12):e1005185, 2016.

[119] Jeffrey S Seely, Matthew T Kaufman, Stephen I Ryu, Krishna V Shenoy, John P Cunningham, and Mark M Churchland. Tensor analysis reveals distinct population structure that parallels the different computational roles of areas m1 and v1. *PLoS Comput Biol*, 12(11):e1005164, 2016.

[120] Arno Onken, Jian K Liu, PP Chamanthi R Karunasekara, Ioannis Delis, Tim Gollisch, and Stefano Panzeri. Using matrix and tensor factorizations for the single-trial analysis of population spike trains. *PLoS computational biology*, 12(11):e1005189, 2016.

[121] Mark M Churchland, M Yu Byron, John P Cunningham, Leo P Sugrue, Marlene R Cohen, Greg S Corrado, William T Newsome, Andrew M Clark, Paymon Hosseini, Benjamin B Scott, et al. Stimulus onset quenches neural variability: a widespread cortical phenomenon. *Nature neuroscience*, 13(3):369, 2010.

[122] Mark M Churchland, John P Cunningham, Matthew T Kaufman, Justin D Foster, Paul Nuyujukian, Stephen I Ryu, and Krishna V Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012.

[123] Patrick T Sadtler, Kristin M Quick, Matthew D Golub, Steven M Chase, Stephen I Ryu, Elizabeth C Tyler-Kabara, M Yu Byron, and Aaron P Batista. Neural constraints on learning. *Nature*, 512(7515):423, 2014.

[124] Jeremy Freeman, Nikita Vladimirov, Takashi Kawashima, Yu Mu, Nicholas J Sofroniew, Davis V Bennett, Joshua Rosen, Chao-Tsung Yang, Loren L Looger, and Misha B Ahrens. Mapping brain activity at scale with cluster computing. *Nature methods*, 11(9):941, 2014.

[125] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[126] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[127] Rahul G Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

[128] Matthew Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems*, pages 2946–2954, 2016.

[129] Sam Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. *Neural computation*, 11(2):305–345, 1999.

[130] Jason Fitzgerald Smith, Kewei Chen, Ajay S Pillai, and Barry Horwitz. Identifying effective connectivity parameters in simulated fmri: a direct comparison of switching linear dynamic system, stochastic dynamic causal, and multivariate autoregressive models. *Frontiers in neuroscience*, 7:70, 2013.

[131] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[132] Gopal Santhanam, Stephen I Ryu, M Yu Byron, Afsheen Afshar, and Krishna V Shenoy. A high-performance brain–computer interface. *nature*, 442(7099):195, 2006.

[133] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[134] Sam T Roweis. Em algorithms for pca and spca. In *Advances in neural information processing systems*, pages 626–632, 1998.

[135] Marlene R Cohen and John HR Maunsell. A neuronal population measure of attention predicts behavioral performance on individual trials. *Journal of Neuroscience*, 30(45):15241–15253, 2010.

[136] David Pfau, Eftychios A Pnevmatikakis, and Liam Paninski. Robust learning of low-dimensional dynamics from large neural ensembles. In *Advances in neural information processing systems*, pages 2391–2399, 2013.

[137] Dmitry Kobak, Wieland Brendel, Christos Constantinidis, Claudia E Feierstein, Adam Kepecs, Zachary F Mainen, Xue-Lian Qi, Ranulfo Romo, Naoshige Uchida, and Christian K Machens. Demixed principal component analysis of neural population data. *Elife*, 5, 2016.

[138] Jeanny Hérault and Bernard Ans. Réseau de neurones à synapses modifiables: Décodage de messages sensoriels composites par apprentissage non supervisé et permanent. *Comptes rendus des séances de l'Académie des sciences. Série 3, Sciences de la vie*, 299(13):525–528, 1984.

[139] Matthias Klemm, Jens Haueisen, and Galina Ivanova. Independent component analysis: comparison of algorithms for the investigation of surface electrical brain activity. *Medical & biological engineering & computing*, 47(4):413–423, 2009.

[140] Simon Daniel Robinson and Veronika Schöpf. Ica of fmri studies: new approaches and cutting edge applications. *Frontiers in human neuroscience*, 7:724, 2013.

[141] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[142] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[143] Christopher Poultney, Sumit Chopra, Yann L Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2007.

[144] H Boulard and Y Kamp. Autoassociative memory by multilayer perceptron and singular values decomposition. *Biol Cybern*, 59:291–294, 1989.

[145] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[146] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 833–840. Omnipress, 2011.

[147] Asif A Ghazanfar and Charles E Schroeder. Is neocortex essentially multisensory? *Trends in cognitive sciences*, 10(6):278–285, 2006.

[148] Rosanna De Meo, Micah M Murray, Stephanie Clarke, and Pawel J Matusz. Top-down control and early multisensory processes: chicken vs. egg. *Frontiers in integrative neuroscience*, 9:17, 2015.

[149] Daniel Christopher Haggerty and Daoyun Ji. Activities of visual cortical and hippocampal neurons co-fluctuate in freely moving rats during spatial behavior. *Elife*, 4, 2015.

[150] Kenneth D Harris and Alexander Thiele. Cortical state and attention. *Nature reviews neuroscience*, 12(9):509, 2011.

[151] Stephan L Marguet and Kenneth D Harris. State-dependent representation of amplitude-modulated noise stimuli in rat auditory cortex. *Journal of Neuroscience*, 31(17):6414–6420, 2011.

[152] Marshall G Shuler and Mark F Bear. Reward timing in the primary visual cortex. *Science*, 311(5767):1606–1609, 2006.

[153] Gonzalo H Otazu, Lung-Hao Tai, Yang Yang, and Anthony M Zador. Engaging in an auditory task suppresses responses in auditory cortex. *Nature neuroscience*, 12(5):646, 2009.

[154] Cristopher M Niell and Michael P Stryker. Modulation of visual responses by behavioral state in mouse visual cortex. *Neuron*, 65(4):472–479, 2010.

[155] Asohan Amarasingham, Stuart Geman, and Matthew T Harrison. Ambiguity and nonidentifiability in the statistical analysis of neural codes. *Proceedings of the National Academy of Sciences*, 112(20):6455–6460, 2015.

[156] Brent Doiron, Ashok Litwin-Kumar, Robert Rosenbaum, Gabriel K Ocker, and Krešimir Josić. The mechanics of state-dependent neural correlations. *Nature neuroscience*, 19(3):383, 2016.

[157] Elad Schneidman, Michael J Berry II, Ronen Segev, and William Bialek. Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature*, 440(7087):1007, 2006.

[158] Michael Vidne, Yashar Ahmadian, Jonathon Shlens, Jonathan W Pillow, Jayant Kulkarni, Alan M Litke, EJ Chichilnisky, Eero Simoncelli, and Liam Paninski. Modeling the impact of common noise inputs on the network activity of retinal ganglion cells. *Journal of computational neuroscience*, 33(1):97–121, 2012.

[159] Malte J Rasch, Arthur Gretton, Yusuke Murayama, Wolfgang Maass, and Nikos K Logothetis. Inferring spike trains from local field potentials. *Journal of neurophysiology*, 99(3):1461–1476, 2008.

[160] John P Cunningham and M Yu Byron. Dimensionality reduction for large-scale neural recordings. *Nature neuroscience*, 17(11):1500, 2014.

[161] Saul Kato, Harris S Kaplan, Tina Schrödel, Susanne Skora, Theodore H Lindsay, Eviatar Yemini, Shawn Lockery, and Manuel Zimmer. Global brain dynamics embed the motor command sequence of caenorhabditis elegans. *Cell*, 163(3):656–669, 2015.

[162] Mark Stopfer, Vivek Jayaraman, and Gilles Laurent. Intensity versus identity coding in an olfactory system. *Neuron*, 39(6):991–1004, 2003.

[163] Urs Köster, Jascha Sohl-Dickstein, Charles M Gray, and Bruno A Olshausen. Modeling higher-order correlations within cortical microcolumns. *PLoS Comput Biol*, 10(7):e1003684, 2014.

[164] Jayant E Kulkarni and Liam Paninski. Common-input models for multiple neural spike-train data. *Network: Computation in Neural Systems*, 18(4):375–407, 2007.

[165] Joao Semedo, Amin Zandvakili, Adam Kohn, Christian K Machens, and M Yu Byron. Extracting latent structure from multiple interacting neural populations. In *Advances in neural information processing systems*, pages 2942–2950, 2014.

[166] Karthik C Lakshmanan, Patrick T Sadtler, Elizabeth C Tyler-Kabara, Aaron P Batista, and M Yu Byron. Extracting low-dimensional latent structure from time series in the presence of delays. *Neural computation*, 2015.

[167] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.

[168] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[169] Martin Vinck, Renata Batista-Brito, Ulf Knoblich, and Jessica A Cardin. Arousal and locomotion make distinct contributions to cortical activity patterns and visual encoding. *Neuron*, 86(3):740–754, 2015.

[170] Wolfram Schultz, Regina M Carelli, and R Mark Wightman. Phasic dopamine signals: from subjective reward value to formal economic utility. *Current opinion in behavioral sciences*, 5:147–154, 2015.

[171] Inge Koch. *Analysis of multivariate and high-dimensional data*, volume 32. Cambridge University Press, 2013.

[172] Ifije E Ohiorhenuan, Ferenc Mechler, Keith P Purpura, Anita M Schmid, Qin Hu, and Jonathan D Victor. Sparse coding and high-order correlations in fine-scale cortical networks. *Nature*, 466(7306):617, 2010.

[173] Peiran Gao and Surya Ganguli. On simplicity and complexity in the brave new world of large-scale neuroscience. *Current opinion in neurobiology*, 32:148–155, 2015.

[174] Nathalie Japkowicz, Stephen Jose Hanson, and Mark A Gluck. Nonlinear autoassociation is not equivalent to pca. *Neural computation*, 12(3):531–545, 2000.

[175] Gopal Santhanam, M Yu Byron, Vikash Gilja, Stephen I Ryu, Afsheen Afshar, Maneesh Sahani, and Krishna V Shenoy. Factor-analysis methods for higher-performance neural prostheses. *Journal of neurophysiology*, 102(2):1315–1330, 2009.

[176] Kazuyuki Hara, Daisuke Saito, and Hayaru Shouno. Analysis of function of rectified linear unit used in deep learning. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE, 2015.

[177] Mark Schmidt. minfunc: unconstrained differentiable multivariate optimization in matlab. *URL https://www.cs.ubc.ca/ schmidtm/Software/minFunc.html*, 2012.

[178] David J Tolhurst, J Anthony Movshon, and Andrew F Dean. The statistical reliability of signals in single neurons in cat and monkey visual cortex. *Vision research*, 23(8):775–785, 1983.

[179] Rufin Vogels, Werner Spileers, and Guy A Orban. The response variability of striate cortical neurons in the behaving monkey. *Experimental brain research*, 77(2):432–436, 1989.

[180] Maoz Shamir and Haim Sompolinsky. Implications of neuronal diversity on population coding. *Neural computation*, 18(8):1951–1986, 2006.

[181] Joel Zylberberg, Alexandre Pouget, Peter E Latham, and Eric Shea-Brown. Robust information propagation through noisy neural circuits. *PLoS computational biology*, 13(4):e1005497, 2017.

[182] Joel Zylberberg. Untuned but not irrelevant: A role for untuned neurons in sensory information coding. *bioRxiv*, page 134379, 2017.

[183] Corbett Bennett, Sergio Arroyo, and Shaul Hestrin. Subthreshold mechanisms underlying state-dependent modulation of visual responses. *Neuron*, 80(2):350–357, 2013.

[184] Pierre-Olivier Polack, Jonathan Friedman, and Peyman Golshani. Cellular mechanisms of brain state–dependent gain modulation in visual cortex. *Nature neuroscience*, 16(9):1331, 2013.

[185] Sinem Erisken, Agne Vaiceliunaite, Ovidiu Jurjut, Matilde Fiorini, Steffen Katzner, and Laura Busse. Effects of locomotion extend throughout the mouse early visual system. *Current Biology*, 24(24):2899–2907, 2014.

[186] Morgane M Roth, Johannes C Dahmen, Dylan R Muir, Fabia Imhof, Francisco J Martini, and Sonja B Hofer. Thalamic nuclei convey diverse contextual information to layer 1 of visual cortex. *Nature neuroscience*, 19(2):299, 2016.

[187] Michel Pierre Janisse. *Pupillometry: The psychology of the pupillary response*. Halsted Press, 1977.

[188] Jacob Reimer, Emmanouil Froudarakis, Cathryn R Cadwell, Dimitri Yatsenko, George H Denfield, and Andreas S Tolias. Pupil fluctuations track fast switching of cortical states during quiet wakefulness. *Neuron*, 84(2):355–362, 2014.

[189] Karolina Socha, Matthew R Whiteway, Daniel A Butts, and Vincent Bonin. Forward visual motion increases arousal and biases tuning measurements in mouse visual cortex. *Current biology*, 2018.

[190] John W Krakauer, Asif A Ghazanfar, Alex Gomez-Marin, Malcolm A MacIver, and David Poeppel. Neuroscience needs behavior: correcting a reductionist bias. *Neuron*, 93(3):480–490, 2017.

[191] Matthew A Smith and Adam Kohn. Spatial and temporal scales of neuronal correlation in primary visual cortex. *Journal of Neuroscience*, 28(48):12591–12603, 2008.

[192] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[193] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[194] Peiran Gao, Eric Trautmann, M Yu Byron, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, and Surya Ganguli. A theory of multineuronal dimensionality, dynamics and measurement. *bioRxiv*, page 214262, 2017.

[195] Ingmar Kanitscheider, Ruben Coen-Cagli, Adam Kohn, and Alexandre Pouget. Measuring fisher information accurately in correlated neural populations. *PLoS computational biology*, 11(6):e1004218, 2015.

[196] Joshua I Glaser, Raeed H Chowdhury, Matthew G Perich, Lee E Miller, and Konrad P Kording. Machine learning for neural decoding. *arXiv preprint arXiv:1708.00909*, 2017.

[197] David BT McMahon, Igor V Bondar, Olusoji AT Afuwape, David C Ide, and David A Leopold. One month in the life of a neuron: longitudinal single-unit electrophysiology in the monkey visual system. *Journal of Neurophysiology*, 112(7):1748–1762, 2014.

[198] Bruno B Averbeck, Peter E Latham, and Alexandre Pouget. Neural correlations, population coding and computation. *Nature reviews neuroscience*, 7(5):358, 2006.

[199] Daniel LK Yamins, Ha Hong, Charles F Cadieu, Ethan A Solomon, Darren Seibert, and James J DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23):8619–8624, 2014.

[200] Ralf M Haefner, Pietro Berkes, and József Fiser. The implications of perception as probabilistic inference for correlated neural variability during behavior. *arXiv preprint arXiv:1409.0257*, 2014.

[201] Klaus Wimmer, Albert Compte, Alex Roxin, Diogo Peixoto, Alfonso Renart, and Jaime De La Rocha. Sensory integration dynamics in a hierarchical network explains choice probabilities in cortical area mt. *Nature communications*, 6:6177, 2015.

[202] Stefano Panzeri, Christopher D Harvey, Eugenio Piasini, Peter E Latham, and Tommaso Fellin. Cracking the neural code for sensory perception by combining statistics, intervention, and behavior. *Neuron*, 93(3):491–507, 2017.

[203] Makoto Fukushima, Richard C Saunders, David A Leopold, Mortimer Mishkin, and Bruno B Averbeck. Differential coding of conspecific vocalizations in the ventral auditory cortical stream. *Journal of Neuroscience*, 34(13):4665–4676, 2014.

[204] Richard O Duda, Peter E Hart, David G Stork, et al. *Pattern classification*, volume 2. Wiley New York, 1973.

[205] Andrew R Mitz, Ramon Bartolo, Richard C Saunders, Philip G Browning, Thomas Talbot, and Bruno B Averbeck. High channel count single-unit recordings from nonhuman primate frontal cortex. *Journal of neuroscience methods*, 289:39–47, 2017.

[206] Avanti Athreya, Donniell E Fishkind, Keith Levin, Vince Lyzinski, Youngser Park, Yichen Qin, Daniel L Sussman, Minh Tang, Joshua T Vogelstein, and Carey E Priebe. Statistical inference on random dot product graphs: a survey. *arXiv preprint arXiv:1709.05454*, 2017.

[207] Daniel J Felleman and DC Essen Van. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex (New York, NY: 1991)*, 1(1):1–47, 1991.

[208] Bo-Cheng Kuo, Mark G Stokes, Alexandra M Murray, and Anna Christina Nobre. Attention biases visual activity in visual short-term memory. *Journal of cognitive neuroscience*, 26(7):1377–1389, 2014.

[209] N Dijkstra, P Zeidman, S Ondobaka, MAJ Gerven, and K Friston. Distinct top-down and bottom-up brain connectivity during visual perception and imagery. *Scientific reports*, 7(1):5677, 2017.

[210] Mark Stokes, Russell Thompson, Rhodri Cusack, and John Duncan. Top-down activation of shape-specific population codes in visual cortex during mental imagery. *Journal of Neuroscience*, 29(5):1565–1572, 2009.

[211] Tomoyasu Horikawa, Masako Tamaki, Yoichi Miyawaki, and Yukiyasu Kamitani. Neural decoding of visual imagery during sleep. *Science*, 340(6132):639–642, 2013.

[212] Yuval Nir and Giulio Tononi. Dreaming and the brain: from phenomenology to neurophysiology. *Trends in cognitive sciences*, 14(2):88–100, 2010.

[213] Paul C Bressloff, Jack D Cowan, Martin Golubitsky, Peter J Thomas, and Matthew C Wiener. What geometric visual hallucinations tell us about the visual cortex. *Neural computation*, 14(3):473–491, 2002.

[214] Renaud Jardri, Kenneth Hugdahl, Matthew Hughes, Jérôme Brunelin, Flavie Waters, Ben Alderson-Day, Dave Smailes, Philipp Sterzer, Philip R Corlett, Pantelis Leptourgos, et al. Are hallucinations due to an imbalance between excitatory and inhibitory influences on the brain? *Schizophrenia bulletin*, 42(5):1124–1134, 2016.

[215] Yanping Huang and Rajesh PN Rao. Predictive coding. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(5):580–593, 2011.

[216] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79, 1999.

[217] Wei Ji Ma, Jeffrey M Beck, Peter E Latham, and Alexandre Pouget. Bayesian inference with probabilistic population codes. *Nature neuroscience*, 9(11):1432, 2006.

[218] Gergő Orbán, Pietro Berkes, József Fiser, and Máté Lengyel. Neural variability and sampling-based probabilistic representations in the visual cortex. *Neuron*, 92(2):530–543, 2016.

[219] Marc O Ernst and Martin S Banks. Humans integrate visual and haptic information in a statistically optimal fashion. *Nature*, 415(6870):429, 2002.

[220] Konrad P Kording and Daniel M Wolpert. Bayesian integration in sensorimotor learning. *Nature*, 427(6971):244, 2004.

[221] Laurence Aitchison and Máté Lengyel. With or without you: predictive coding and bayesian inference in the brain. *Current opinion in neurobiology*, 46:219–227, 2017.

[222] Bruno A Olshausen. 20 years of learning about vision: Questions answered, questions unanswered, and questions not yet asked. In *20 Years of Computational Neuroscience*, pages 243–270. Springer, 2013.

[223] Eric Jonas and Konrad Paul Kording. Could a neuroscientist understand a microprocessor? *PLoS computational biology*, 13(1):e1005268, 2017.

[224] Yuri Lazebnik. Can a biologist fix a radio?or, what i learned while studying apoptosis. *Cancer cell*, 2(3):179–182, 2002.

[225] Joshua T Vogelstein, Katrin Amunts, Andreas Andreou, Dora Angelaki, Giorgio Ascoli, Cori Bargmann, Randal Burns, Corrado Cali, Frances Chance, Miyoung Chun, et al. Grand challenges for global brain sciences. *arXiv preprint arXiv:1608.06548*, 2016.

[226] Larry F Abbott, Dora E Angelaki, Matteo Carandini, Anne K Churchland, Yang Dan, Peter Dayan, Sophie Deneve, Ila Fiete, Surya Ganguli, Kenneth D Harris, et al. An international laboratory for systems and computational neuroscience. *Neuron*, 96(6):1213–1218, 2017.