Building theories of neural circuits with machine learning

Sean Robert Bittner

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

COLUMBIA UNIVERSITY

2021

Abstract

Building theories of neural circuits with machine learning

Sean Robert Bittner

As theoretical neuroscience has grown as a field, machine learning techniques have played an increasingly important role in the development and evaluation of theories of neural computation. Today, machine learning is used in a variety of neuroscientific contexts from statistical inference to neural network training to normative modeling. This dissertation introduces machine learning techniques for use across the various domains of theoretical neuroscience, and the application of these techniques to build theories of neural circuits.

First, we introduce a variety of optimization techniques for normative modeling of neural activity, which were used to evaluate theories of primary motor cortex (M1) and supplementary motor area (SMA). Specifically, neural responses during a cycling task performed by monkeys displayed distinctive dynamical geometries, which motivated hypotheses of how these geometries conferred computational properties necessary for the robust production of cyclic movements. By using normative optimization techniques to predict neural responses encoding muscle activity while ascribing to an “untangled” geometry, we found that minimal tangling was an accurate model of M1. Analyses with trajectory constrained RNNs showed that such an organization of M1 neural activity confers noise robustness, and that minimally “divergent” trajectories in SMA enable the tracking of contextual factors.

In the remainder of the dissertation, we focus on the introduction and application of deep generative modeling techniques for theoretical neuroscience. Specifically, both techniques employ recent advancements in approaches to deep generative modeling – normalizing flows – to capture complex parametric structure in neural models. The first technique, which is designed for statistical generative models, enables look-up inference in intractable exponential family models. The efficiency of this technique is demonstrated by inferring neural firing rates in a log-gaussian poisson model of spiking responses to drift gratings in primary visual cortex. The second technique is designed for statistical inference in mechanistic models, where the inferred parameter distribution is constrained to produce emergent properties of computation. Once fit, the deep generative model confers analytic tools for quantifying the parametric structure giving rise to emergent properties. This technique was used for novel scientific insight into the nature of neuron-type variability in primary visual cortex and of distinct connectivity regimes of rapid task switching in superior colliculus.

Table of Contents

List of Figures

List of Algorithms

Acknowledgements

First and foremost, thank you to my parents Robert and Kimberly Bittner for their love and support, and for always encouraging me to pursue my interests. My PhD at the Center for Theoretical Neuroscience at Columbia has been an extraordinary experience, which exceeded even my lofty expectations. I am appreciative to so many people who have supported me and helped me grow since I joined the program. In general, it has been a treat to be part of such an impressive, curious, and genuine group of people.

Thanks to Jelena Kovačević, who mentored me in my first undergraduate research project along with Siheng Chen at Carnegie Mellon University. Thank you to Byron Yu for introducing me to a field that I became so passionate about, taking the time to help me develop from an engineer into a scientist, and for giving me so many opportunities to advance my career. Thank you to everyone from the Center for the Neural Basis of Cognition in Pittsburgh that I had the priviledge to meet or work with during this period: Benjamin Cowley, Ryan Williamson, William Bishop, Matthew Golub, Akash Umakantha, Jay Hennig, Joao Semedo, Adam Snyder, Brent Doiron, Matthew Smith, and Steven Chase.

Thank you to Mark Churchland for being such a positive and supportive mentor, for teaching me a lot about motor control and theoretical neuroscience, and especially for allowing me to do such interesting rotation work in his lab with Abby Russo, which turned into a special project. Thank you to Abby for all of the advice and guidance she's given me throughout graduate school. Much of my foundational understanding of neural dynamics was developed at Mark's journal club, and I am grateful to all friends and colleagues that I met in this group: Cora Ames, Karen

Schroeder, Najja Marshall, Andrew Zimnik, Sean Perkins, and Yana Pavlova.

Thank you to all of John Cunningham's current and former advisees that I have overlapped with. You have all been amazing friends, mentors, and zebroid compatriots. Gamaleldin Elsayed, Gabriel Loaiza-Ganem, Scott Linderman, Shreya Saxena, Josh Glaser, Anqi Wu, Geoff Pleiss, Taiga Abe, Kelly Buchanan, Dan Biderman, Julien Boussard, Luhuan Wu, and Elliot Rodriguez.

Thanks to all friends and colleagues I have met through the theory center: Jacob Portes, Ramin Khajeh, Matthias Christenson, James Priestly, Dar Gilboa, Ella Batty, Patrick Stinson, Salomon Muller, Chris Cueva, Conor Dempsey, Grace Lindsay, Amin Nejatbakhsh, Marjorie Xie, Danil Tyulmankov, Denis Turcu, Fabio Stefanini, Alessandro Ingrosso, Laureline Logiac, James Murray, Chris Rodgers, SueYuen Chen, Vivek Athalye, Rainer Engelken, Bettina Hein, Kenny Kay, Jackson Loper, Ramon Nogueira Manas, Samuel Muscinelli, Ari Pakman, Xuexin Wei Erdem Varol, Matthew Whiteway, Robert Yang, Ashok Litwin-Kumar, Sean Escola, Allison Ong, and Mahham Fayyaz. A special thanks goes to Jeff Seely, who was an incredibly helpful mentor during my early PhD.

Thank you to all of my friends from the Neurobiology and Behavior PhD program for the comradery and support, especially Gabe Stine, Jung Park, Leslie Sibener, Kelly Martyniuk, Naveen Senhilnathan, Luke Nunnelly, Avery McGuirt, Josie McGowan, Lillian Coie, Viktoriya Zhuravleva, Jennifer Scribner, Josie McGowan, Lisa Randolph, Macayla Donegan, Georgia Pierce, Rebeca Vaadia, Michelle Stackman, Liz Pekarskaya, Gordon Petty, Michael Cohanpour, Claire Everrett, Andres Villegas, and Stevie Hamilton. Thank you to Rozanna Yakub and Alla Kerzhner for supporting all of us.

Thanks to my collaborators Agostina Palmigiano, Alex Piet, Ann Duan, and their mentors Kenneth Miller and Carlos Brody.

Thanks to my thesis committee members for providing their insight and support: Larry Abbott, Kenneth Miller, Liam Paninski, and external member Jonathan Pillow.

Thanks to my adviser John Cunningham, who has been an amazing mentor. I'm very grateful for the impact he's had on my growth as an academic, and for his ability to listen and help me clarify my career ambitions. Even after several years, I am still impressed by his ingenuity and willingness to explore unorthodox ideas, which has made my PhD all the more enjoyable.

Dedication

To Mom, Dad, Brian, Jake, and Dan.

Chapter 1: Introduction and Background

To understand the neural basis of behavior and cognition, neuroscientists have conducted extensive experiments and studies to describe neurons and their circuitry, cellular mechanisms, and genetic factors [1, 2]. To build on these achievements, theoretical neuroscientists seek a lawful, integrated understanding of neural computation through the perspective of mathematical modeling [3, 4, 5]. Mathematical models of neural computation are constrained by the biological and physiological properties of the nervous system being studied. While single neuronal models can provide important insight (e.g. [6, 7]), the predominant focus of theoretical neuroscience is dedicated towards neural circuit modeling: networks of interconnected neurons that make up a given brain area or subcircuit [8, 9, 10, 11, 12].

As the field of theoretical neuroscience has progressed, so has the field of machine learning. Leaps in the capabilities of image and speech recognition were enabled by neural network architectures [13, 14], whose origins come from neuroscientific motivations [15]. Key ideas in each of these fields have aided and advanced the other throughout their co-evolution [16, 17, 18]. Machine learning has become an integral part of theoretical neuroscience, and the specific technique employed in a given study depends on the type of theory being evaluated.

Theories of neural computation can be categorized by the questions they attempt to answer: *What* are neurons doing? *How* are they doing it? *Why* do they do it that way [19]? In this introductory chapter, we explain the different types of theories, the types of models they stipulate, and the machine learning techniques used to evaluate such theories. With this context, we then introduce the technique of deep generative modeling, which is a core topic of this dissertation. Finally, we review the content of this dissertation, which introduces new machine learning techniques for theoretical neuroscience, and develops theories of neural circuits through the application of such techniques.

1.1 Types of theories in neuroscience

Here, we recapitulate the organization of theories in neuroscience by Levenstein et al. 2020 [19], which will serve as useful categories for explaining the role of machine learning in theoretical neuroscience. *Descriptive* theories in neuroscience seek to explain *what* a neural circuit does. For example, such theories often describe the stimulus features [20], behavior [21], or abstract representations [22] that neurons are responsive to. *Mechanistic* theories aim to explain *how* key elements and properties of the neural circuit enable it to perform its computation. The mathematical structure of a mechanistic model reflects the biological constraints of the neural circuit; this is how neurophysiological research informs theoretical modeling [9]. *Normative* theories aim to explain *why* a neural circuit exhibits some phenomena. Within some established constraints of the normative model, a neural circuit is proposed to optimize some criteria. For example, normative models are implicit throughout neuroscience – auditory cortex is considered to be optimized for processing sound. In the next section, we describe how descriptive, mechanistic, and normative theories are evaluated with a variety of models, and how various machine learning techniques are used to support this science.

1.2 Models and machine learning techniques in theoretical neuroscience

Different types of theories prescribe different types of models, which are mathematical formalizations of theories. Such models are used to ask empirical questions: Does the theory explain observed phenomena (or data)? What predictions does the theory make? When analytic techniques cannot answer these questions, techniques from the domain of machine learning are often used to fit or train these models. In this chapter, we review the primary classes of models used in theoretical neuroscience and the machine learning techniques used to analyze them.

1.2.1 Statistical generative models

To learn about the brain from neural recordings, neuroscientists have adopted statistical modeling techniques to make the most of collected data. Descriptive theories (see types of theory in Section 1.1) are typically embedded in such statistical generative models, for which considerable methodology for neural data has been developed [23]. Neural responses are recorded with respect to a particular stimulus or behavior (often in the same repeated condition), and by using statistical methodology, one can precisely quantify what the data informs us about a neuron or neuronal population.

Statistical inference – the inference of model parameters most likely to produce observed data – is the core machine learning technique developed for research with these models. Seminal work modeling neural spiking data as point processes produced methods for inferring neural firing rate [24] and a neuron's relation to extrinsic experimental factors [25]. Advanced techniques supported this inference in concert with spiking history from the remaining ensemble [26, 27], facilitating the evaluation of a mechanistic theory that the network connectivity supports the hypothesized representation.

State-space modeling of neural firing rates has grown in popularity, where unobserved factors are inferred from the shared firing rate variability of the neural population [28, 29, 30]. By inferring the low dimensional state of neural population activity, scientists can infer how computations are executed through internal dynamics [31]. In this case of state space modeling of neural activity, a mechanistic theory (at the level of dynamics predicated by neural connectivity) is built from the low dimensional state inferred using a phenomenological model.

A major advance in the machine learning community for statistical inference was the development of the variational autoencoder (VAE) [32, 33], which has had a sizable impact on the field of statistical neuroscience. VAEs use deep neural networks to induce a posterior distribution on hidden variables of a latent variable model given data. So far, VAEs have been used to expand the class of generative models of cortical population activity [34, 35, 36, 37] and animal behavior [38, 39, 40] amenable to statistical inference. We will revisit the topic of VAEs when we introduce the topic of deep generative modeling in Section 1.3.

1.2.2 Neural circuit models

Neural circuit models are constructed via a system of mathematical equations to evaluate mechanistic theories of neural computation. The equations of neural circuit models often reflect the constraints of the underlying biology, and are designed to reproduce phenomena observed from the neural system being modeled. Neural circuit models come in a variety of sizes, structures, and levels of abstraction. For example, prominent models of fly and crustacean subcircuits may consist of five or less neurons [41, 42], while models of cortical areas in other model organisms may be on the scale of hundreds [43] or thousands [44]. Models range from the scope of specific microcircuits [45], to entire cortical areas [46], to whole brain models [47]. The computation modeled may be related to sensory processing, action, or some intermediate processing.

In biologically realistic neural circuit models, the variables are often neural activity level (e.g. membrane potential or firing rate), and the parameters of the model equations (e.g. ion channel conductance, time constants, synaptic strength, or connectivity rate) govern the evolution of this activity. For example, a model of the stomatogastric ganglion (STG) of crustaceans consists of five neurons: two in the fast population (f1 and f2), two in the slow population (s1 and s2), and a hub neuron that is both synaptically and electrically coupled to each population [41] (Fig. 1.1 top-left). The synaptic and electrical conductance parameters g_{synA}, g_{synB}, and g_{el} determine how the membrane potentials of each neuron in this circuit evolve according to well-established biophysical laws of cellular dynamics in neurons [48]. For different values of the conductance parameters, the hub neuron's spiking frequency will either sync with the fast, slow, or both populations. Since the STG model contains neurons corresponding to real, identifiable cells, and it closely reflects biophysical laws at the level of channel conductances, we consider it to have a high degree of biological realism. Furthermore, since the coupling organization of this neural circuit model cannot be unwound into a purely feed-forward structure, it is a *recurrent* neural circuit model.

Not all biologically realistic neural circuit models have recurrent architectures. For example,

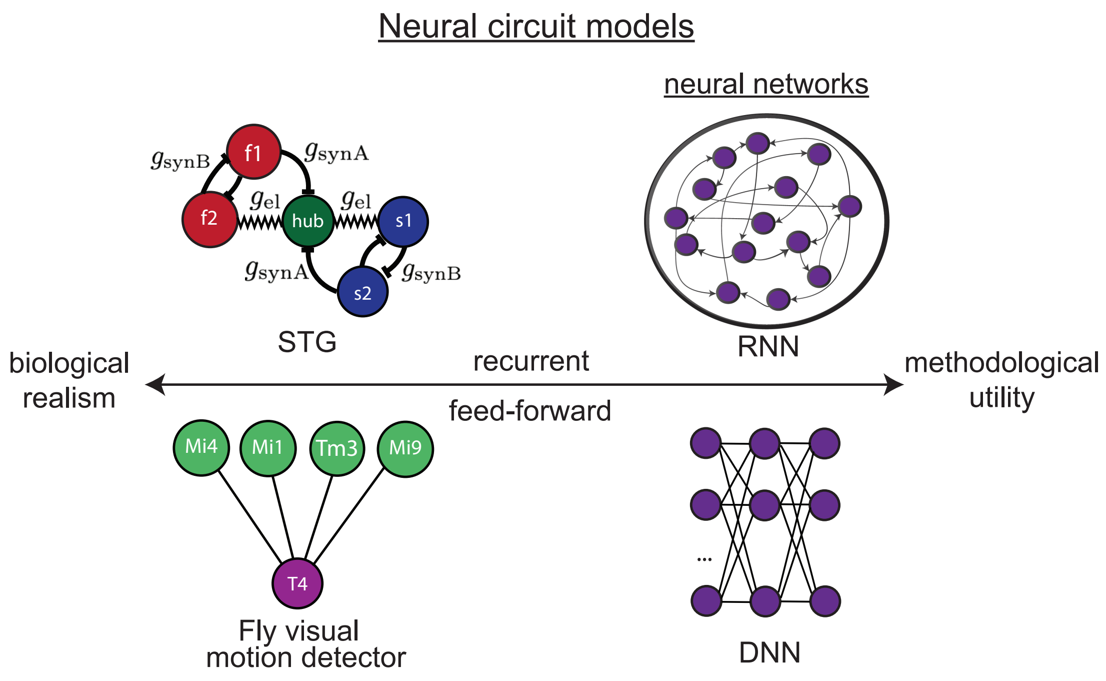


Figure 1.1: Neural circuit models. Examples are organized by biological realism (left), machine learning methodological utility (right), recurrent (top), and feed-forward architecture (bottom). STG - stomatogastric ganglion subcircuit of crustaceans to model hub neuron coupling [41] (see text). Fly visual motion detector - three-neuron microcircuit of *Drosophila* that detects directional motion [42] (see text). RNN - recurrent neural network. DNN - deep neural network.

the motion detector subcircuit in the fly *Drosophila* contains four neurons (Mi4, Mi1, Tm3, and Mi9), which project to the detector neuron T4 [42] (Fig. 1.1 bottom-left). By closely characterizing the input to these medullar cells and the strength and dynamics of their projections to T4, scientists attempt to reverse engineer the precise algorithm implemented for motion detection [49]. In contrast to recurrency, this architecture is classified as a *feed-forward* circuit, since neurons Mi4, Mi1, Tm3, and Mi9 are not influenced by the neural activity of T4.

It is often methodologically challenging to find the parameters of biologically realistic models that reproduce observed phenomena. In Section 1.2.1, we described the technique of statistical inference, which often does not generalize to this class of models because of the absence of a tractable likelihood function. To infer the parameters of such neural circuit models, simulation-based inference is used, where the model is simulated many times until suitable parameters are obtained [50]. Historically, neuroscientists have used the sampling techniques of approximate bayesian computation [51] and sequential monte carlo [52] to fit (or "invert") neural circuit models.

In the next section, we introduce the more abstract subclass of neural circuit models, which we classify as *neural network* models. We will consider neural networks as belonging to the class of neural circuit models when they are used with the intention of modeling a computation executed by a biological neural system. Otherwise, a neural network is *not* considered a neural circuit model, and merely a utilitarian function approximator used to enable machine learning methodology.

1.2.3 Neural networks

The origin of the neural network architecture is rooted in the McCullough-Pitts neuron [15], which is designed to integrate the incoming signals from neurons projecting incoming synapses. Unsupervised [53] and supervised [54] learning rules for such neural networks were developed in the mid-twentieth century, however the implementation of efficient backpropagation optimization techniques for neural networks did not occur until the 1980's [55, 56, 57]. Throughout the next four decades, the generality and flexibility of neural networks coupled with efficient optimization technology would make deep learning the dominant paradigm of machine learning [16]. No matter the application or problem domain, neural networks have proven to be highly effective and tractable function approximators.

Neural networks belong to two main classes of architectures: recurrent (Fig. 1.1 top-right) and feed-forward (Fig. 1.1 bottom-right). Recurrent neural networks (RNNs) produce outputs as a map from an internal hidden neural state which evolves over time via some internal connectivity and the input it receives. RNNs are typically used to model sequence data. Feed-forward or deep neural networks (DNNs), which consist of a sequence of nonlinear transformations at each layer of neural units, are used to learn static mappings. In their everyday usage in machine learning technology, we do not consider these neural networks as neural circuit models, but utilitarian classes of functions. Only when they are used in a context of neural system modeling, do we consider them to be neural circuit models.

For example, a popular approach in theoretical neuroscience has been to train an RNN to perform some task, and to reverse engineer how the RNN executes the computation through its dynamics [58]. The machine learning techniques of backpropagation and backpropagation-through-time [59, 60] are used to train neural networks, whether it be in the context of unsupervised, supervised, or reinforcement learning [18]. Furthermore, some research tests the efficacy of biologically plausible learning rules in RNNs [61, 62, 63, 64]. In these examples, we would consider such RNNs to be neural circuit models. The impressive performance of convolutional deep learning architectures in computer vision [13] suggest that visual cortices may process visual stimuli in a similar manner. Research has related the (approximately) feed-forward visual stream of primates to individual components of the deep learning architecture [65]. In this case as well, the deep neural network would be considered as a neural circuit model designed to evaluate a mechanistic theory.

1.2.4 Normative models

Normative models of neural activity suggest that it is optimal with respect to some criteria while adhering to some constraints. These constraints may be biologically realistic in the spirit of a mechanistic theory, or they may be more abstract and conceptual. For example, the gabor-like responses of neurons in primary visual cortex can be explained by a normative model: gabor responses emerge when synthetic neural activity are optimized to encode natural visual stimuli under the constraint of a sparse linear code [66]. Machine learning techniques for constrained optimization and regularization are often required to fit such normative models.

1.3 Deep generative modeling

In Section 1.2.1, we introduced statistical generative models, and how they can be used to approximate distributions of data. By inferring the parameters most likely to generate a given dataset, we also gain the ability to produce samples with variation according to the learned distribution. In deep generative models, deep neural networks are used to define the generative model, and generally increase the expressivity of the approximating model class. Deep generative models are usually fit using VAEs [32] or generative adversarial networks (GANs) [67].

The typical architecture of a deep generative model consists of a non-invertible deterministic mapping of a latent variable to the mean and variance parameters of a generating gaussian (or other) distribution. Recent advances in deep generative modeling have produced a new class of deep generative models called *normalizing flows*, which are parameterized by *invertible* neural network architectures [68, 69]. Special care is taken to design deep, expressive architectures that remain invertible and either produce fast samples [70], fast probability density calculations [71], or both [72].

Normalizing flows consist of two key components: a simple random initial distribution, and a deep neural network. The simple random initial distribution (typically chosen as an isotropic gaussian) has no parameters and is the source of randomness in the probability distribution. This initial randomness is then deterministically transformed by the deep neural network to capture the distributional structure of interest. The deep neural networks of normalizing flows are constrained to be bijective, and to have tractable log determinant jacobians. As a note, normalizing flows should not be misconstrued as neural circuit models, since they are simply representing a distribution which is flexibly parameterized by the neural network. Continued research has resulted in a host of powerful techniques for approximating distributions with rich structure [69].

1.4 Thesis overview

In this dissertation we introduce machine learning techniques for theoretical neuroscience, and use these methods to develop theories of neural circuits. In Chapter 2, we develop optimization techniques and train RNNs to evaluate normative theories of motor cortex based on the dynamical geometry of neural responses to movement. In the remainder of the thesis, we introduce deep generative modeling techniques for building descriptive and mechanistic theories. Both techniques introduced in Chapters 3 and 4 employ normalizing flows to capture rich structure in parameter distributions of neural models. In Chapter 3, we present an efficient method for posterior inference with normalizing flows in statistical generative models (for descriptive theories), which we use for fast inference in a log-gaussian poisson model of neural spiking responses in primary visual cortex. In Chapter 4, we present a method that uses normalizing flows for inference in mechanistic neural circuit models. Finally in Chapter 5, EPI is used to develop mechanistic theories of neuron-type variability in primary visual cortex, and of superior colliculus connectivity regimes producing rapid task switching.

Chapter 2: The dynamical geometry of population activity in motor cortex

In this chapter, we evaluate normative theories of primary motor cortex (M1) and supplementary motor area (SMA) based on the dynamical geometry of neural responses observed during a cycling task performed by monkeys. Neural responses in M1 and SMA displayed cyclical and helical geometries, repectively, motivating the development of metrics to quantify dynamical geometry of such nature. Such geometries were theorized to confer noise robustness and in the case of SMA the ability to track of contextual factors. We developed an optimization-based technique for evaluating a variety of normative models of M1 activity. Futhermore, we trained trajectory-constrained RNNs to produce responses from both M1 and SMA to evaluate how dynamical geometric properties confer the robust production of cyclical movements in the presence of noise.

This work is featured across two studies presenting minimal "tangling" as a normative model of M1 [73] and minimal "divergence" as a normative model of SMA [74]. Sections 2.2-2.2.6 are lightly adapted from Russo et al. 2018 and were coauthored by Abigail A. Russo, Sean M. Perkins, Jeffrey S. Seely, Brian M. London, Antonio H. Lara, Andrew Miri, Najja J. Marshall, Adam Kohn, Thomas M. Jessell, Laurence F. Abbott, John P. Cunningham and Mark M. Churchland. Sections 2.3.1-2.3.3 are lightly adapted from Russo et al. 2020 and were coauthored by Abigail A. Russo, Ramin Khajeh, Sean M. Perkins, John P. Cunningham, L. F. Abbott, and Mark M. Churchland. Section 2.4 takes from both papers and describes the analyses performed in this chapter. All remaining text provides summary of motivation and experimental results from the two studies supporting the modeling analyses.

2.1 Introduction

Motor cortex, which has synaptic projections to both spinal interneurons [75] and motoneurons [76], has been shown to strongly represent both movement-related kinematics [21, 77, 78] and muscle activity [79, 80, 81]. However, it has also been shown that neural responses contain features reflecting network or feedback dynamics [82, 83]. Research into the neural basis of motor control has largely focused on analyzing neural responses from reaching tasks, and has produced little consensus on how to account for the variety of these phenomena. To interrogate the nature of movement encoding and recurrent dynamics in motor cortex, Russo et al. [73] designed a novel behavioral paradigm in which a hand pedal is rotated for different cycle counts in a forward and backward direction (Fig. 2.1A-B). By collecting motion (Fig. 2.1C-D), neural (Fig. 2.1C) and electromyographic (EMG) recordings (Fig. 2.1D) from monkeys during this behavior, kinematic, neural, and muscle activity could be compared over extended periods of time.

During the execution of this task, individual neurons had heterogeneous responses properties. At the population level, neurons coded for muscle activity quite well (R^2 = 0.79), yet the population response was dominated by a rotational component that did not code for muscle or kinematic activity. This most salient feature of the population recordings motivated the precise characterization of dynamic population-level geometry in motor cortex. Russo et al. found that recordings in primary motor cortex (M1) [73] and supplementary motor area (SMA) [74] revealed distinctive population geometries (minimal tangling and divergence, respectively). In our analyses, we evaluate how well such geometrical measures confer hypothesized computational properties, and whether they define accurate normative models of recorded neural activity.

2.2 Motor cortex embeds commands in an untangled population response

2.2.1 Smooth dynamics predict low tangling

Recent physiological and theoretical investigations suggest that the neural state in motor cortex obeys smooth dynamics [82, 84, 85, 86, 87]. Smooth dynamics imply that neural trajectories

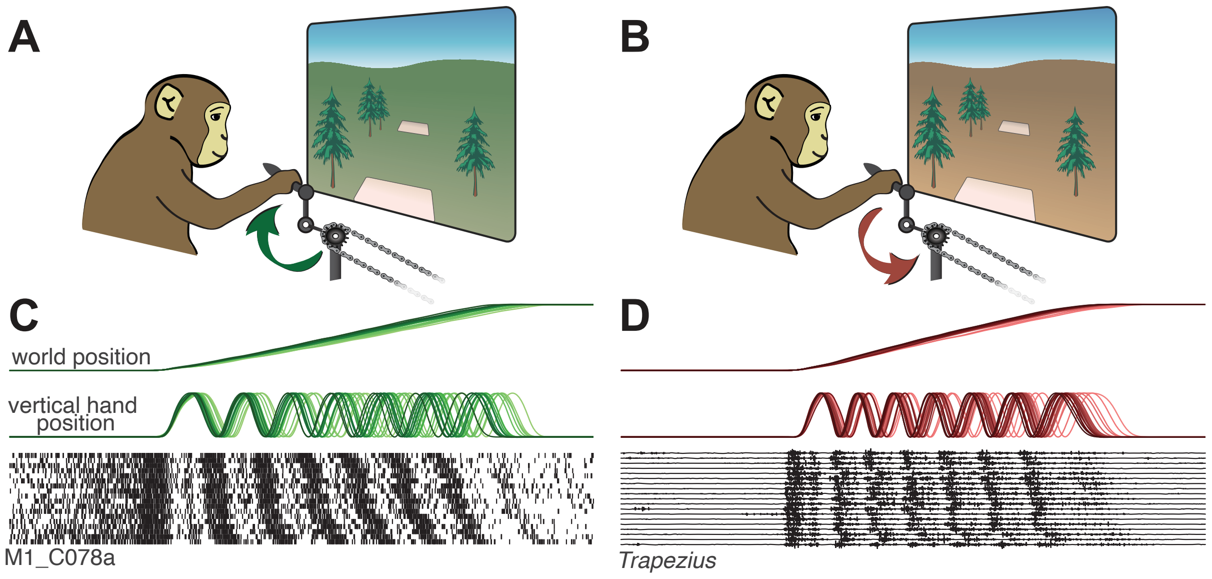


Figure 2.1: Behavioral and physiological responses during cycling. **A**. Schematic of the task during forward cycling. A green landscape indicated that virtual progress required cycling “forward". **B**. An orange landscape indicated that progress required cycling “backward". **C**. Behavioral data and spikes from one neuron during an example session. Data are for a single condition: forward / seven-cycle / bottom-start (monkey C). Trials are aligned to movement onset, and ordered from fastest to slowest. **D**. Behavioral data and raw trapezius EMG for one condition: backward / seven-cycle / bottom-start (monkey D).

should not be “tangled": similar neural states, either during different movements or at different times for the same movement, should not be associated with different derivatives. We quantified trajectory tangling using

$$Q(t) = \max_{t'} \frac{||\dot{\mathbf{x}}_t - \dot{\mathbf{x}}_{t'}||^2}{||\mathbf{x}_t - \mathbf{x}_{t'}||^2 + \epsilon} \tag{2.1}$$

where $\mathbf{x}_t$ is the neural state at time t (i.e., a vector containing the neural responses at that time), $\dot{\mathbf{x}}_t$ is the temporal derivative of the neural state, $||\cdot||$ is the Euclidean norm, and ϵ is a small constant that prevents division by zero (methods). $Q(t)$ becomes high if there exists a state at a different time, t', that is similar but associated with a dissimilar derivative. We take the maximum to ask whether the state at time t ever becomes tangled with any other state. This maximum is taken with t indexing across time during all conditions. $Q(t)$ can be analogously assessed for the muscle trajectories.

We chose tangling as a straightforward measure of whether a given trajectory could have been produced by a smooth dynamical flow-field. Given limits on how non-smooth dynamics can be, moments of very high tangling are incompatible with a fixed flow-field. Furthermore, even moderately high tangling implies potential instabilities in the underlying flow-field (Supp Fig 1 and Supp Note of [73]). High tangling thus implies that the system must rely on external commands rather than internal dynamics, or that the system is flirting with instability. Although other metrics are possible, tangling has the practical benefit that it can be computed directly from the trajectories without needing to know (or fit) a flow-field.

For the reasons above, a network that relies heavily on intrinsic dynamics should avoid tangling. In contrast, when population activity primarily reflects external commands (as for the muscles or a population of sensory neurons) high tangling is both benign and, with enough observations, likely. For example, co-contraction of the biceps and triceps at one moment might need to be quickly followed by biceps activation and triceps relaxation. At a later moment or during a different movement, co-contraction might instead need to be followed by biceps relaxation and triceps activation. This would constitute an instance of tangling because the same state (co-contraction) is followed by different subsequent states. Do such moments of high tangling indeed occur for the muscles? If so, are they mirrored or avoided in the neural responses?

Russo et al. [73] showed that neural activity (in recordings and network models) had less tangling than the muscle activity it produced, suggesting that similar neural states with conflicting derivatives are avoided by motor cortex. Lower tangling of motor cortex than muscle population activity was also found in monkeys performing reaching tasks and in mice performing reaching and walking behaviors, suggesting that untangled motor responses are preserved across tasks and species. In contrast, somatosensory cortex and visual cortex did not display the untangling property.

2.2.2 Noise-robust networks display low tangling

For a recurrent or feedback-driven network, it is intuitive that high tangling must be avoided. If the flow-field has some degree of smoothness, nearby states cannot be associated with very different derivatives. Thus, moments of high tangling cannot be produced without relying on disambiguating external inputs. Yet motor cortex trajectories avoided even moderate tangling. This is not strictly necessary even in the idealized case of a fully autonomous dynamical system. For example, some recurrent networks did show moderate tangling, yet still functioned [73]. Might the very low empirical tangling confer some computational advantage? Formal considerations support that possibility: even moderate tangling implies potential dynamical instabilities (Supp Note of [73]).

To explore potential advantages of low tangling, we considered neural networks trained to generate a simple idealized output: $\cos t$ for one muscle and $\sin 2t$ for a second muscle (Fig 2.2A, top). The resulting output trajectory was thus a figure-eight (left sub-panel). It is not possible for a network's internal trajectory to follow a pure figure-eight; the center-most state is very highly tangled. Tangling can be reduced by employing a third dimension such that the trajectory is: $[\cos t; \sin 2t; \beta \sin t]$. Even a modest value of β reduces tangling enough (middle sub-panel) that the trajectory can be produced. As a network follows that three-dimensional trajectory, the figure-eight trajectory can still be "read out" via projection onto two of the axes (with the third dimension falling in the null space of the readout [88, 89]. Is there an advantage to further decreases in tan-

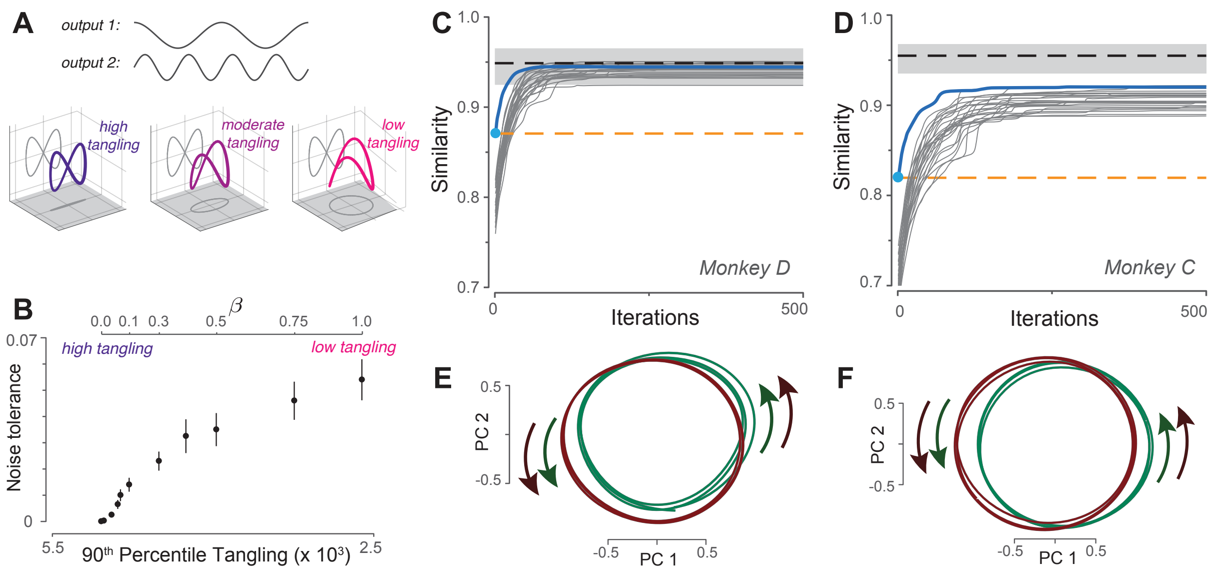


Figure 2.2: Low trajectory tangling aids noise robustness, and can be leveraged to predict the motor-cortex population response. **A**. Illustration of how an output can be embedded in a larger trajectory with varying degrees of tangling. Top gray traces: A hypothetical two-dimensional output $[\cos t, \sin 2t]$. Plotted in state space, the output trajectory is a figure-eight, and contains a highly tangled central point. Adding a third dimension $(\beta \sin t)$ reduces tangling. B. Noise robustness of recurrent networks trained to follow the internal trajectory $[\cos t, \sin 2t, \beta \sin t]$ By varying β, we trained a set networks that could all produce the same output, but had varying degrees of trajectory tangling. Noise tolerance (mean and SEM across initializations) is plotted versus network tangling for each value of β. C. Similarity of the predicted and empirical motor-cortex population responses (monkey D). Blue trace: prediction yielded by optimizing the cost function in Eqn. 2. Cyan dot indicates similarity at initialization; i.e., the similarity of empirical neural and muscle trajectories. This also provides a lower benchmark (orange dashed line). Gray traces: Same as blue trace but initialized with Gaussian noise added during initialization. Multiple initializations were yielded a family of predictions. Black dashed line shows upper benchmark as described in the text, with a 95% confidence interval computed across random divisions of the population. D. Same but for monkey C. E. Projection of a representative predicted population response onto the top two PCs. Prediction based on EMG for monkey D. Green / red traces show trajectories for three cycles of forward / backward cycling. F. Same but for monkey C. See also Fig 2.3 and 2.4.

gling (right sub-panel)? We examined noise tolerance across networks whose internal trajectories were $[\cos t; \sin 2t; \beta \sin t]$ with different values of β. This necessitated the unusual step of training networks not only to produce a desired output, but also to follow a specified internal trajectory (see Section 2.4.1).

Networks with high trajectory tangling failed to produce the figure-eight output trajectory in the presence of even small amounts of noise (Fig 2.2B). Networks with low trajectory tangling were much more noise robust. We performed a similar analysis with trajectories that encoded the empirical muscle trajectories, but with varying degrees of tangling (found using the optimization approach in the next section). Again, low tangling provided noise robustness (Fig. 2.3). This was true both for networks that generated a single internal trajectory, and networks that generated different “forward" and “backward" trajectories based on inputs. Intuitively, when tangling is low it is less likely that noise will perturb the network onto a nearby but inappropriate part of the trajectory. More formally, low tangling aids local stability (Supp Fig 1; Supp Note of [73]). While the example in Fig 2.2A,B is intentionally simplified, it illustrates a feature that may help interpret the empirical neural trajectories. Note that $\beta = 1$ yields a weakly-tangled trajectory that encodes the desired figure-eight output in one projection and is a circle in another projection (Fig 2.2A, right sub-panel). Although we created this shape via construction, it is a natural shape to introduce: a circle is the least-tangled rhythmic trajectory.

2.2.3 Hypothesis-based prediction of neural responses

The results above suggest a hypothesis: motor cortex may embed outgoing commands (which, if muscle-like, would be quite tangled) in a larger trajectory such that the full orbit is minimally tangled. Inspired by optimizations that successfully predicted V1 responses [66], we employed an optimization approach to predict the dominant patterns of motor cortex activity. Optimization found a predicted neural population response, $\hat{X}$, that could be linearly decoded to produce the

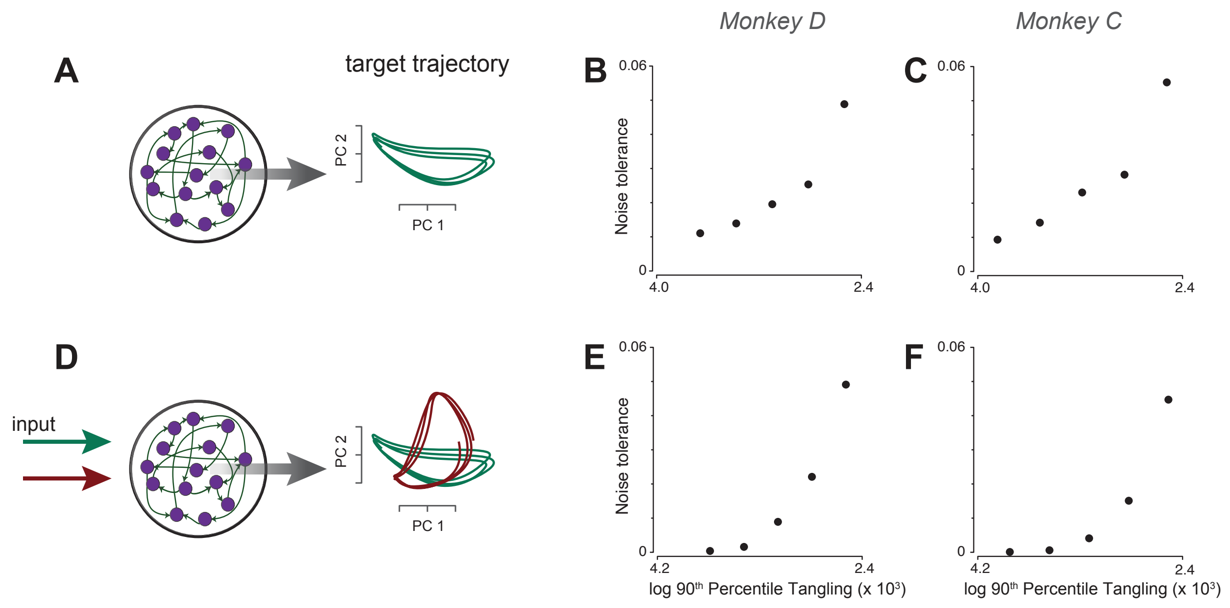


Figure 2.3: Relationship between low tangling and noise robustness in networks trained to follow specified internal trajectories. These trajectories encoded muscle activity with varying degrees of tangling. **A**. Schematic of network architecture and internal trajectory for networks trained to produce trajectories corresponding to forward cycling only. Networks (50 fully connected units) were trained to produce ten-dimensional target trajectories that encode muscle activity with varying degrees of trajectory tangling. To create target trajectories, we used an optimization that was the same as that described in the main text (and that produced the data in Figure 7C-F) but was applied to a single cycle of muscle data for forward cycling only. Optimization was repeated 10 times with smooth noise added during initialization to produce a family of solutions. As optimization ran, we kept the solution for different iterations: 0, 1, 2, 3, 4, 5, 10 ,100, and the final iteration. This yielded 90 trajectories: one for each optimization and iteration. These trajectories were all ten-dimensional and had a wide variety of tangling values. For each such trajectory, 20 networks (each with a different set of initial weights) were trained to autonomously and repeatedly follow that trajectory. As for Figure 7B, networks were not trained to produce the trajectory as an output but rather to internally follow that trajectory. **B**, **C**. Analysis of the noise robustness of the networks described in A. Noise tolerance was assessed by training networks in the presence of different levels of additive Gaussian noise. Noise tolerance was defined as the maximum noise level at which the network still followed the target trajectory. Each black circle plots the mean noise tolerance across many networks whose tangling fell within a given bin. Standard errors are within the symbol size. **D**. Schematic for networks trained to produce trajectories corresponding to either forward or backward cycling depending on an input. The input was two-dimensional. The command to produce forward / backward cycling involved one dimension being high and the other low. Each input dimension was connected to all network units with random weights. All other details are as in A. **E**, **F**. Same as B,C, but for the networks described in D.

empirical muscle activity Z, yet was minimally tangled. Specifically:

$$\hat{X} = \underset{X}{\operatorname{argmin}} \left(||Z - ZX^+X||_F^2 + \lambda \sum_t Q_x(t) \right) \tag{2.2}$$

where each column of the matrix Z describes the muscle population response for one time and condition. The first term of the cost function ensures that neural activity “encodes" muscle activity; ZX^+X is the optimal linear reconstruction of Z from X (+ indicates the pseudo-inverse; $||\cdot||_F$ indicates the Frobenius norm). This formulation should not be taken to imply that the true neural-to-muscle mapping is linear, merely that the predicted neural activity should yield a reasonable linear readout of muscle activity, consistent with empirical findings [90, 91, 92]. The second term of the cost function encourages low trajectory tangling. The predicted neural population response thus balances optimal encoding of muscle activity with minimal tangling.

We applied optimization using muscle data that included three middle cycles of forward cycling and three middle cycles of backward cycling. Thus, we are attempting to simultaneously predict two “steady state" neural trajectories. We used canonical correlation to assess the similarity between predicted and actual neural responses. Canonical correlation finds linear transformations of two datasets such that they are maximally correlated. We employed a variant of canonical correlation that enforces orthonormal matrix transformations. Unity similarity thus indicates two datasets are the same but for a rotation, isotropic scaling, or offset. We initialized optimization with $\hat{X}_{\text{init}} = Z$, corresponding to the baseline hypothesis that neural activity is a “pure" code for muscle activity. This resulted in a reasonably high initial similarity (Fig 2.2C-D, cyan dot) because muscle activity shares many basic features with neural activity (e.g., the same fundamental frequency).

During optimization, we insisted that the predicted neural population response, $\hat{X}$, have the same dimensionality as the muscle population response, Z (both were ten-dimensional). Matching dimensionality is a conservative choice that aids interpretation. Because optimization cannot add dimensions, some muscle-like features must be lost in order to gain features that reduce tangling. Similarity will therefore increase only if the features gained during optimization are more realistic

/ prominent than the features that are lost. Similarity between predicted and empirical populations increased with optimization (Fig 2.2C-D blue), reaching a similarity roughly halfway between the “pure muscle encoding" hypothesis and perfect similarity. To provide a rough benchmark of good similarity, we computed the average similarity between two random halves of the empirical neural population (black dashed trace with 95% confidence intervals). Similarity approached this benchmark for both monkeys. To test the consistency of this result we repeated optimization, each time initializing with the empirical patterns of muscle activity plus temporally smooth noise in each of the ten dimensions. Similarity to the data always increased (gray traces). This analysis also revealed that the addition of random structure decreased initial similarity (gray traces start below the blue trace). This underscores that increasing similarity requires the addition of structure matching that in the neural data, rather than any arbitrary structure.

Each initialization resulted in a slightly different solution (the optimized $\hat{X}$). We were thus able to ask which solutions were common and whether the nature of those solutions explains the increased similarity with the empirical data. For all 200 solutions (100 per monkey), optimization produced near-circular trajectories. When comparing between forward and backward, two classes of solution emerged. The less common (31/100 for monkey D and 13/100 for monkey C) involved dominant circular trajectories in planes that were nearly orthogonal (first principal angle > 85°) for forward and backward. The most common (69/100 and 87/100 for monkey D and C) involved at least some overlap between these planes. In such cases, trajectories were almost always co-rotational (67/69 and 85/87 for monkey D and C) in the top two PCs. Two typical solutions are shown in Figure 2.2E,F. Co-rotations dominate because, when two trajectories exist in a common subspace, tangling is lowest if they co-rotate (if they exist in orthogonal planes, co-rotation versus counter-rotation is not defined). Similar structure was seen for the empirical data: the planes that best captured neural trajectories during forward and backward cycling overlapped (principal angles were 72° and 61° for monkey D, and 73° and 40° for monkey C) and showed co-rotation in the top two PCs. Thus, the hypothesis embodied in Equation 2.2 not only increased quantitative similarity, it also reproduced the dominant features of the neural data: nearly circular trajectories that exist in distinct but overlapping planes, and that co-rotate in the projection capturing the most variance.

2.2.4 Alternative predictions

We performed a variety of optimizations corresponding to cost functions embodying other hypotheses (Fig 2.4). Optimizations that sought to reduce the norm of activity or to increase sparseness (standard forms of regularization) led to decreases in similarity. Optimizing for local smoothness (one aspect of low tangling) increased similarity but not as much as optimizing for low tangling itself. Thus, similarity increased only when optimization reduced tangling, and increased most when low tangling was directly optimized.

However, low tangling per se was not necessarily sufficient to increase similarity. We created simulated populations where the response of each unit was either the response of a muscle or the derivative of that response. This reflects the hypothesis that neurons might represent both muscle activity and the change in muscle activity [93]. By construction, these simulated populations had fairly low tangling [73]. Yet, they did not particularly resemble the neural population. Quantitatively, similarity increased modestly for monkey D (roughly half as much as when optimizing for low tangling directly) and decreased for monkey C. The dominant signals in these simulated populations also did not show the same dominant circular structure seen in the neural data [73]. The mismatch can be understood by noting that differentiation increases the prevalence of high-frequency features. This does not lead to a match with the dominant circular structure at the fundamental frequency in the empirical data. In summary, optimizing directly for low tangling introduced features that were both particularly effective in reducing tangling and matched features in the data. Reducing tangling in a more “incidental" fashion did not produce these realistic features.

2.2.5 Signals introduced by optimization yield incidental correlations

The optimization based on Equation 2.2 added structure that reduced tangling. That structure is unconnected to kinematics or other task parameters; optimization was blind to all such parameters. Nevertheless, the predicted neural population response appeared to encode kinematics to a greater

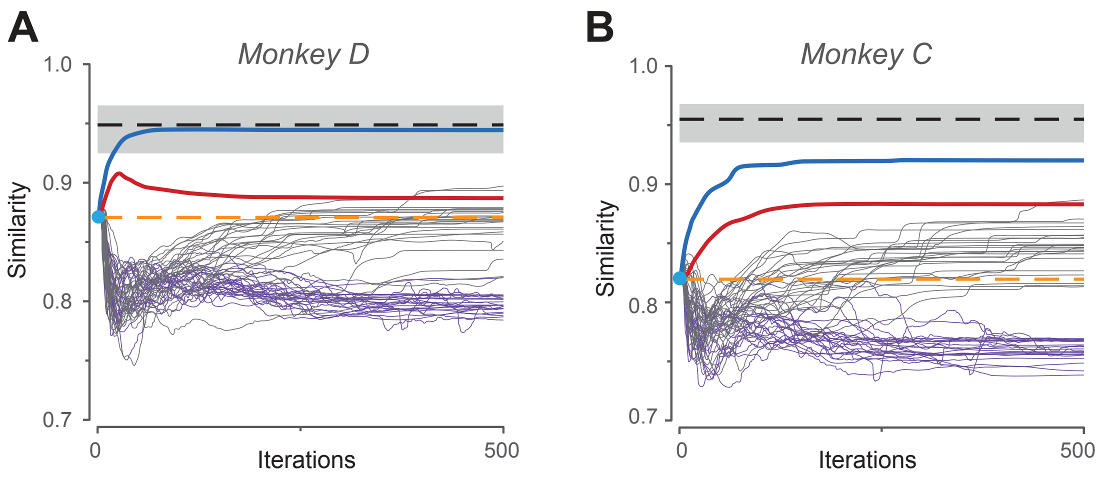


Figure 2.4: Elaboration of analyses in Figure 2.2C,D **A**, **B**. Same as Figure 2.2C,D but using additional cost functions. These cost functions are described below, and formalized subsequently. Each cost function embodies a hypothesis regarding the relationship between neural and muscle activity. The similarity metric thus indicates how well that hypothesis predicts the data. Blue traces (reproduced from Figure 2.2) show similarity between empirical and predicted population responses when prediction employed the cost function in Equation 2.2. That cost function included linear-decode error and trajectory tangling. Optimization thus embodies the hypothesis that neural activity seeks to encode muscle activity fairly directly while maintaining low tangling. Purple traces: predictions yielded by minimizing non-linear decode error and the L2-norm of population activity. Optimization thus embodies the hypothesis that neural activity may wish to be as modest as possible while still allowing muscle activity to be decoded. Each muscle was allowed its own non-linearity, the parameters of which were optimized. This potentially allowed neural activity to be lower-dimensional and/or simpler than muscle activity, with different patterns of activity across muscles accounted for via different non-linearities. In principle, this might have explained why the dominant neural signals are 'simpler' and different from the dominant muscle signals. In fact, similarity between the empirical and predicted populations typically declined. (There were many local minima so the algorithm was run from many different initializations.) Gray traces: predictions yielded by minimizing both non-linear decode error and trajectory tangling. This cost function embodies the same hypothesis as in Equation 2.2, but allows each muscle's activity to be decoded nonlinearly as above. Across multiple initializations, similarity occasionally increased, especially when compared to the purple traces. However, similarity did not increase to the same degree as for the simpler cost function in Equation 2.2. This might mean that the 'true' readout is already close to linear (such that the constraint of linearity is beneficial). More likely, the space of non-linear readouts is sufficiently large that we did not find an instance where the non-linear model improved upon the linear approximation. Red trace: prediction yielded by minimizing linear-decode error and trajectory curvature within each condition. Trajectory curvature is effectively a local measure of tangling. Similarity increased, but not as much as if tangling was minimized directly. Not shown: prediction yielded by minimizing linear-decode error and sparseness. Similarity declined dramatically and immediately, with traces falling off the bottom of the plot.

degree than would a pure code for muscle activity. We used linear regression to decode a set of kinematic parameters (horizontal and vertical position and velocity) from the activity of the muscle population. Fits were reasonable (R^2=0.86 and 0.88 for monkey D and C) but improved (R^2=0.97 and 0.94) when we instead decoded kinematics from the predicted neural population response. This performance was nearly identical to that observed when decoding kinematics from the empirical neural population (R^2=0.98 and 0.93). The ability to decode horizontal and vertical velocity might initially seem surprising: the dominant signals in the neural data co-rotated in the top two PCs – inconsistent with a velocity representation. However, the presence of more than two dimensions with sinusoidal structure ensured that velocity could be read out reasonably accurately. Despite these excellent decodes, generalization performance was poor: generalization R^2 was near-zero (or even negative) when fitting kinematics for one direction and predicting for the other. This was true whether decoding was based on the predicted or empirical neural response. While poor generalization does not exclude the possibility that the empirical population encodes kinematic signals, we saw no direct evidence for this hypothesis. As noted above, we also rarely observed neurons whose firing rates resembled kinematic parameters.

2.2.6 Muscle-like signals are embedded in trajectories with low tangling

The optimization results lead to the hypothesis that the dominant population-level signals in motor cortex function to yield low tangling, and that muscle-like signals may be encoded by relatively modest 'ripples' in dimensions that point off the plane of dominant circular structure. A rough analogy would be a phonograph, where the direction that encodes a temporally complex output is orthogonal to the dominant motion of the record. Can such structure be viewed directly in the empirical data? We projected the neural population response onto triplets of dimensions (Fig 2.5). The first and second dimensions were always the first two PCs. The third was based on the readout direction of a particular muscle, defined by the set of weights found via linear regression (arrow in Fig 2.5A plots the readout direction for the trapezius). The third dimension was then the vector that was orthogonal to the first two PCs, and allowed the three dimensions to span the readout direction.

Consider first a triplet of dimensions that span the trapezius readout direction (Fig 2.5A). Trajectories trace out circular paths in the top PCs. Ripples in a third dimension yield the fine temporal structure that matches trapezius activity Fig 2.5B). The overall trajectory thus has the joint properties of encoding trapezius activity while exhibiting low tangling. Similar structure was observed for other muscles (Fig 2.5C,E).

The dimensions that encode muscle activity captured only modest variance. In the examples in Figure 2.5, each muscle-readout dimension captured 10% as much variance as each of the top two PCs. The vertical dimensions in 2.5A,C,E are thus shown on an expanded scale for visualization. Similar structure was present for the network models and also for the predicted population responses in Figure 2.2E,F: the activity of each "encoded" muscle constituted a set of ripples upon dominant circular structure that yielded low tangling.

In addition to the dimensions from which muscle-like signals can be read out, there exist other dimensions (not visible in Figure 2.5) that provide separation between neural trajectories during forward and backward cycling. Low tangling may require such separation, else forward and backward trajectories would have to encode very different patterns of muscle activity despite following similar paths. Indeed, forward and backward neural trajectories were on average much better separated than the corresponding muscle trajectories. This difference in separation was large but not as profound as the difference in tangling. Thus, low neural-trajectory tangling (relative to muscle-trajectory tangling) results from a variety of factors: more circular trajectories, increased separation between forward and backward trajectories, and greater alignment of flow-fields (e.g., co-rotation in the dominant dimensions).

2.3 Supplementary motor area exhibits a minimally divergent geometry

Here, we consider the hypothesis that supplementary motor area (SMA) guides movement by tracking contextual factors, and derive a prediction regarding population trajectory geometry. We predict that SMA trajectories should avoid 'divergence'; trajectories should be structured, across

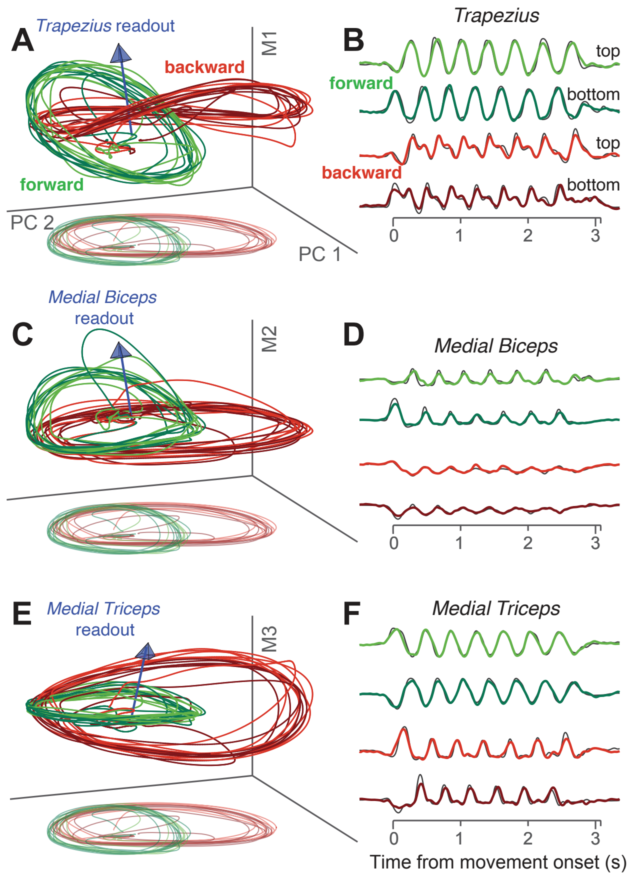


Figure 2.5: Muscle-like signals coexist with signals that contribute to low tangling. Data are for monkey D. **A**. Three-dimensional subspace capturing trajectories that encode trapezius activity; i.e., can be linearly read out to approximate trapezius activity. Blue arrow indicates the readout direction, defined by the weights identified via linear regression. Axes correspond to the first two PCs and a third dimension that ensures the space spans the readout direction. Trajectories are shown for four conditions: forward (green) and backward (red) seven-cycle movements, starting at the top and bottom (lighter and darker traces). Lighter 'shadow' traces at bottom show the projection onto just the first two PCs (perspective has been added). **B**. Projections, for the four conditions plotted in **A**, onto the readout direction. Thin black trace plots the true activity of the trapezius. Axis spans the time of movement. **C**,**D**. Same as A,B but for the medial biceps. Only the third (vertical) axis is different. **E**,**F**. Same but for the medial triceps.

time and conditions, such that it is never the case that two trajectories follow the same path and then separate. Low trajectory divergence is essential to ensure that neural activity can distinguish situations with different future motor outputs, even if current motor output is similar. We hypothesize that the need to avoid divergence strongly influences the shape of the population trajectory, and thus the response features observed within a particular task.

2.3.1 Trajectory divergence

Trajectories displayed by context-tracking networks reflect specific solutions to a general problem: ensuring that two trajectory segments never trace the same path and then diverge. Avoiding such divergence is critical when network activity must distinguish between situations that have the same present motor output but different future outputs. Rather than assessing the specific paths of individual-network solutions, we developed a general metric of trajectory divergence. We note that trajectory divergence differs from trajectory tangling [73], which was very low in both SMA and M1 (Figure S5 of [74]). Trajectory tangling assesses whether trajectories are consistent with a locally smooth flow-field. Trajectory divergence assesses whether similar paths eventually separate, smoothly or otherwise. A trajectory can have low tangling but high divergence, or vice versa (Figure 2.7).

To construct a metric of trajectory divergence, we consider times t and t', associated population states x_t and $x_{t'}$, and future population states $x_{t+\Delta}$ and $x_{t'+\Delta}$. We consider all possible pairings of t and t' across both times and cycling distances. Thus, t and t' might occur during different cycles of the same movement or during different distances. We compute the ratio $\frac{|x_{t+\Delta}-x_{t'+\Delta}|^2}{|x_t-x_{t'}|^2+\alpha}$, which becomes large if $x_{t+\Delta}$ differs from $x_{t'+\Delta}$ despite x_t and $x_{t'}$ being similar. The constant α is small and proportional to the variance of x, and prevents hyperbolic growth.

Given that the difference between two random states is typically sizeable, the above ratio will be small for most values of t'. As we are interested in whether the ratio ever becomes large, we take the maximum, and define divergence for time t as:

$$D(t) = \max_{t',\Delta} \frac{|x_{t+\Delta} - x_{t'+\Delta}|^2}{|x_t - x_{t'}|^2 + \alpha} \tag{2.3}$$

We consider only positive values of Δ. Thus, $D(t)$ becomes large if similar trajectories diverge but not if dissimilar trajectories converge. Divergence was assessed using a twelve-dimensional neural state. Results were similar for all reasonable choices of dimensionality.

$D(t)$ differentiated between context-tracking and context-naive networks. To compare, we considered pairs of networks, one context-tracking and one context-naïve. For each time, we plotted D(t) for the context-tracking network versus that for the context-naive network. Trajectory divergence was consistently lower for context-tracking networks (Figure 6C of [74], p<0.0001, rank sum test). This was further confirmed by considering the difference in $D(t)$ for every time and all network pairs (Figure 6D of [74]). Both context-tracking and context-naïve trajectories contained many moments when divergence was low, resulting in a narrow peak near zero. However, context-naive trajectories (but not context-tracking trajectories) also contained moments when divergence was high, yielding a large set of negative differences.

2.3.2 Computational implications of trajectory divergence

We considered trajectory divergence because of its expected computational implications. A network with a high-divergence trajectory can accurately and robustly generate its output on short timescales. Yet unless guided by external inputs at key moments, such a network may be susceptible to errors on longer timescales. For example, if a trajectory approximately repeats, a likely error would be the generation of extra cycles or the inappropriate skipping of a cycle.

To test whether these intuitions are accurate, we performed additional simulations. We employed an atypical training approach that enforced an internal network trajectory [73], as opposed to the usual approach of training a target output. We trained networks to precisely follow the empirical M1 trajectory, recorded during a four-cycle movement, without any input indicating when to stop (Figure 2.6A). To ensure that solutions were not overly delicate, networks were trained in the presence of additive noise. Using data from each monkey, we trained forty networks: ten for each of the four four-cycle conditions. Networks were able to reproduce the cyclic portion of the M1 trajectory. However, without the benefit of a stopping pulse, these networks failed to consistently complete the trajectory. For example, networks sometimes erroneously produced extra cycles (Figure 2.6B) or skipped cycles and stopped early (Figure 2.6C).

We also trained networks to follow the empirical SMA trajectories. Those trajectories contained both a rhythmic component and lower-frequency 'ramping' signals (Figure 2.6D) related to the translation visible in Figure 4C,D. In contrast to the high-divergence M1 trajectories, which were never consistently followed for the full trajectory, the majority of network initializations resulted in good solutions where the low-divergence SMA trajectory was successfully followed from beginning to end. Thus, in the absence of a stopping pulse, the empirical SMA trajectories could be reliably produced and terminated in a way that the M1 trajectories could not.

2.3.3 Discussion

A variety of studies argue that SMA contributes to the guidance of action based on internal, abstract, or contextual factors [94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106]. We translated this hypothesis into a prediction regarding the geometry of population activity, and tested it in a novel task. As predicted, trajectory divergence was low in SMA, and provided a cohesive explanation for diverse response features. Slowly ramping firing-rates are, at the surface level, a very different feature from changes in the occupied subspace. Yet both contribute to low divergence. Other features (which we did not attempt to isolate) maintained low divergence across cycling directions and starting positions. This raises a broader point: the features that subserve low divergence will almost certainly be task and situation specific. For example, during sequences of reaches, SMA neurons exhibit burst-like responses with various forms of selectivity. Such selectivity presumably produces low divergence, although this remains to be explicitly tested. Thus, a reasonable hypothesis is that, during a given task, SMA responses will exhibit some of the dom-

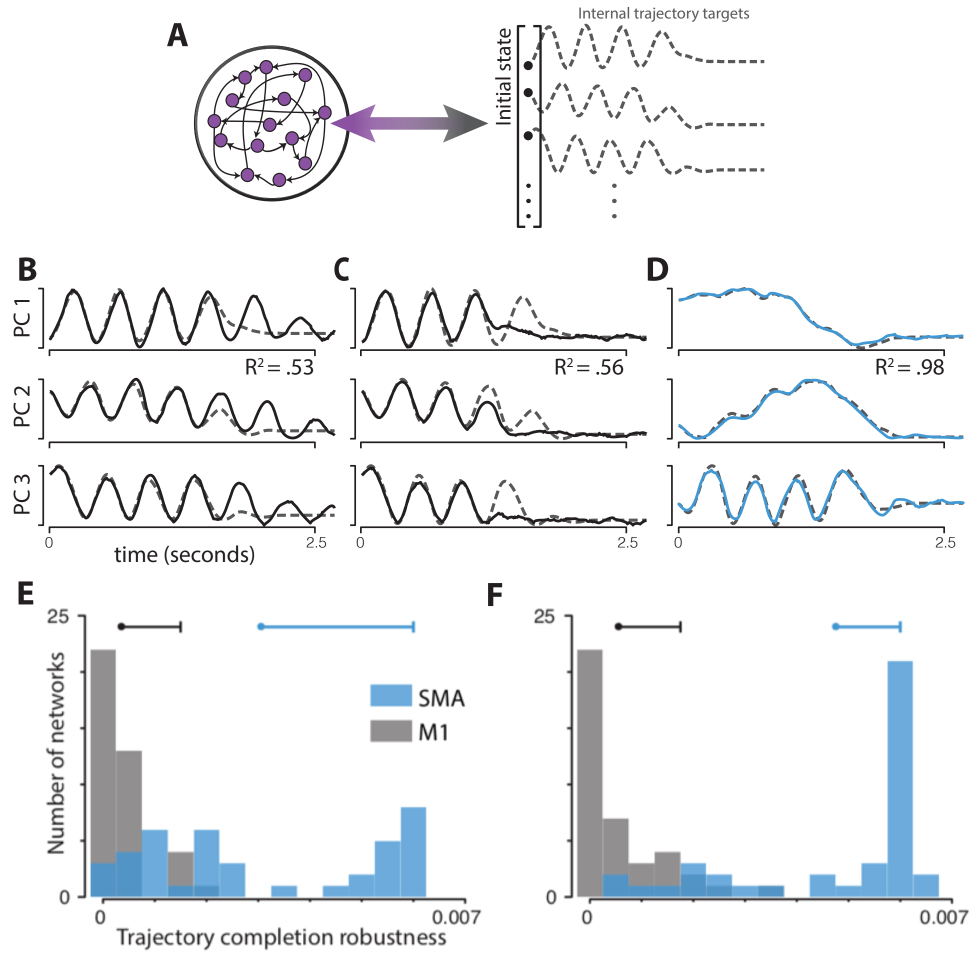


Figure 2.6: **A**. Illustration of trajectory-constrained neural networks. Networks were trained to autonomously follow a target trajectory defined by the top six PCs of the empirical population trajectory during a four-cycle movement, including stopping at the end. Dashed lines show the target trajectory for three PCs for one example: monkey D, M1, cycling backward starting at the bottom. The activity of every neuron in the network was trained to follow a random combination of the projection onto the top six PCs. This ensured that the simulated population trajectory matched the empirical trajectory. **B**, **C**. Two example network trajectories (black lines) constrained to follow M1 target trajectory (dashed gray lines) during a 4-cycle condition. These networks were less noise robust than those following the SMA target trajectory and tended to produce too many cycles (B) or abort early (C). **D**. An example network trajectory (blue lines) constrained to follow SMA target trajectory (dashed gray lines). **E**. Trajectory completion robustness of networks constrained to follow either the M1 (gray) or SMA (blue) population trajectories during the 4-cycle conditions (monkey C). 10 networks were trained for each of the four 4-cycle conditions (all combinations of starting position and pedaling direction) for each region. Dots correspond the mean of each distribution and rightward-going hash corresponds to the 90th percentiles. **F**. Same for monkey D.

inant response features seen in M1 (transient responses when reaching, rhythmic activity during cycling, and so forth) combined with additional response features that ensure low divergence. It is common for studies to focus on specific features that relate to how a network might perform a particular task or computation [82, 107, 108, 89, 109, 99, 110]. This will remain an essential strategy. A complementary strategy is to quantify general properties likely to be preserved across a class of computations. Our divergence metric was designed with this goal in mind. We recently considered a different geometric property, trajectory tangling [73], which is necessary for a network to robustly generate an output via internal dynamics. Low trajectory tangling was observed in M1 across a range of tasks, in both monkeys and mice. As another example, studies of the visual system have employed linear separability (a different definition of "untangled") to assess whether population geometry is consistent with a class of computation having been performed [111, 112]. The advantages of this approach come with a limitation: geometry may strongly suggest a class of computations, yet do little to delineate the specific computation. For example, low trajectory divergence in SMA is consistent with internal tracking of context, but does not specify the input-output relationship the network is trying to accomplish. Indeed, we observed low-divergence trajectories regardless of whether context-tracking networks received a ramping input or internally generated their own ramp. Similarly, it remains unclear what signals SMA conveys to downstream areas. Possibilities include start/stop signals, a ‘keep moving’ signal that remains high during movement, or a rhythmic signal that entrains downstream pattern generation [113]. Deciphering the computation used to perform a particular task will typically require consideration of a level of detail below that captured by measures of population geometry. A goal of assessing population geometry is to find properties that generalize across situations. At the same time, exceptions may be informative. For example, during grasping, trajectory tangling becomes high in M1, suggesting a shift in the balance of input-driven versus internally driven activity [114]. We expect that, in SMA, there will be situations where divergence becomes revealingly high. For example, there are presumably limits on the timescales across which SMA can track context, which may be revealed in the timescales over which divergence stays low. Trajectory divergence is also likely to become high when action is guided by sudden, unpredictable cues. Given the benefits of low divergence, why employ separate areas – SMA and M1 – with low and high trajectory divergence? Why not unify context tracking and pattern generation? Allowing high divergence in M1 may be useful for two reasons. First, dispensing with divergence-avoiding signals frees dynamic range for other computations, such as generating fine-grained aspects of the outgoing motor command. Second, low divergence may interfere with adaptation; learning on one cycle would have no clear way of transferring to other cycles if they involve very different neural states [115]. The concepts in the present study are informed by our field's understanding of how recurrent networks perform computations [109, 86, 99, 73, 116]. Because recurrent-network-based computations are commonly described via flow-fields governing a neural state [117, 58], this perspective has been termed a "dynamical systems view" [83]. This view intersects with ideas regarding how dynamical systems can perform computations [118] or describe behavior [119]. It has been argued that dynamics-based explanations should supplant 'representational' explanations [118]. This view is extreme – dynamical systems may involve representations [120] – yet it is true that purely representational thinking can be limiting. For example, the question of whether M1 is more concerned with "muscles versus movements" is poorly addressed by inquiring whether neural activity is a function of muscle activity versus movement kinematics[121, 73, 122]. M1 activity is dominated by signals that are neither muscle-like nor kinematic-like, but are readily understood as necessary for low trajectory tangling and thus for noise-robust dynamics [73]. Correspondingly, we found multiple properties of the SMA population response that can be understood as aiding low trajectory divergence. It is tempting to apply representational interpretations to some of those properties. For example, there is a dimension in which activity is ramp-like during cycling, which might be thought of as a representation of 'time', 'distance', or 'progress within the overall movement'. While it is conceivable that this dimension might consistently represent these things during other tasks, there is presently no evidence for this. Furthermore, low divergence is aided by additional features that lack a straightforward representational interpretation, such as the occupancy of different subspaces across cycles. The dynamical perspective helps one to see the connection between these seemingly

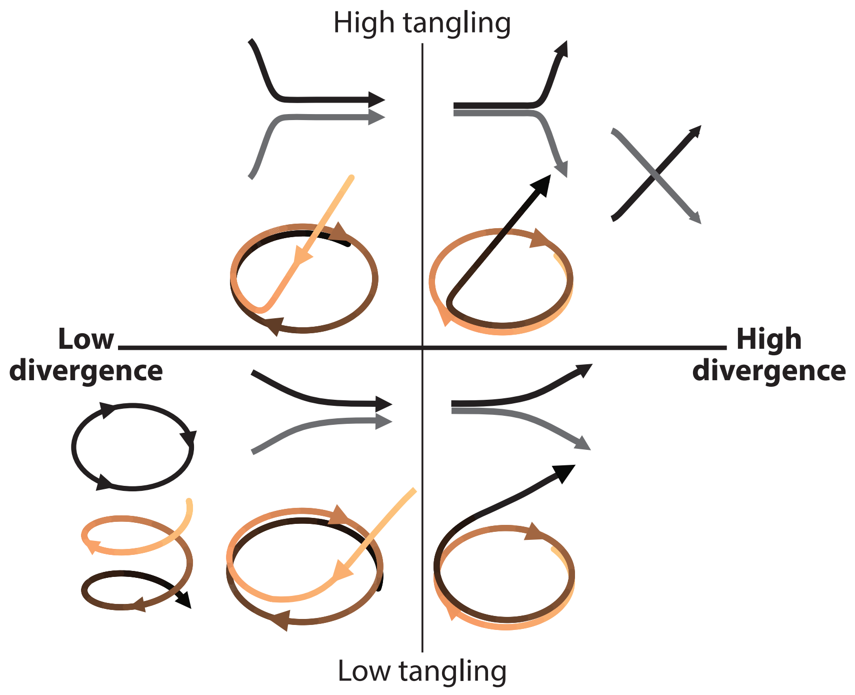


Figure 2.7: Illustration of trajectories that would yield low or high trajectory divergence and trajectory tangling. Pairs of lines (black and gray) indicate trajectories that might correspond to two different conditions while circular tan-black lines indicate trajectories that might correspond to a single condition over time. Trajectories that have high tangling (upper two quadrants) may have sharp turns and crossing points. Trajectories that have high divergence (right two quadrants) are similar at some point in time but later separate. Divergence will remain low (left two quadrants) if trajectories start dissimilar and converge (e.g. trajectories in the right column), start similar and stay similar (e.g. black circular trajectory in the bottom left quadrant), or maintain dissimilarity over time (e.g. helical trajectory at the bottom left corner).

disjoint response features, in a way that a purely representational perspective does not.

2.4 Methods

2.4.1 Trajectory-constrained Neural Networks

To examine how tangling relates to noise-robustness (Figure 2.7B) we trained RNNs to follow a set of target internal trajectories. This involved the unconventional approach of employing both a target output, $\mathbf{y}_{\text{targ}}$, and a target internal network trajectory, $\mathbf{s}_{\text{targ}}$. Networks consisted of 100 units.

Network dynamics were governed by

$$\mathbf{v}(t+1) = \mathbf{v}(t) + \frac{\Delta t}{\tau}\left(-\mathbf{v}(t) + Af(\mathbf{v}(t)) + \mathbf{w}(t)\right) \tag{2.4}$$

$$\mathbf{y}(t) = Cf(\mathbf{v}(t)) \tag{2.5}$$

where $f :=$ tanh and $\mathbf{w} \sim \mathcal{N}(0, \sigma_w I)$ adds noise. $\mathbf{v}$ can be thought of as the membrane voltage and $f(\mathbf{v}(t))$ as the firing rate. $Af(\mathbf{v}(t))$ is then the network input to each unit: the firing rates weighted by the connection strengths. $Cf(\mathbf{v}(t))$ is a linear readout of firing rates.

During training, A was adjusted using recursive least squares [62] so that $Af(\mathbf{v}(t) \approx \mathbf{s}_{\text{targ}}$. Training thus insured that the synaptic inputs to each unit closely followed the pre-determined trajectory defined by $\mathbf{s}_{\text{targ}}$. Firing rates therefore also followed a pre-determined trajectory. C was adjusted so that $\mathbf{y} \approx \mathbf{y}_{\text{targ}}$. Training was deemed successful if the R^2 between $\mathbf{y}$ and $\mathbf{y}_{\text{targ}}$ was > 0.9. Noise tolerance was assessed as the largest value of σ_w for which the network could be trained to accurately produce the target output for five consecutive cycles (R^2>0.9 between $\mathbf{y}$ and $\mathbf{y}_{\text{targ}}$, averaged across 100 iterations) despite the constraint of following the target internal trajectory, $\mathbf{s}_{\text{targ}}$.

We set $\mathbf{y}_{\text{targ}} = [\cos t, \sin 2t]$. To construct $\mathbf{s}_{\text{targ}}$, we began with an idealized low-dimensional target, $\mathbf{s}'(t)_{\text{targ}} = \left[\cos t, \sin 2t, \beta \sin t\right]$. To give each unit a target, we set $\mathbf{s}_{\text{targ}} = G\mathbf{s}'_{\text{targ}}$ where G is a random matrix of size 100×3 with entries drawn independently from a uniform distribution from -1 to 1. Noise tolerance was tested for a range of values of β. That range produced target trajectories that varied greatly in their tangling, allowing us to examine how tangling related to noise tolerance. Noise tolerance was the largest magnitude of state noise for which the network still produced the desired output. For each target trajectory, and each of the 20 random initializations of A, C, and G, we doubled σ_w starting at 0.005 until we found the noise tolerance. We then computed the average (and SEM) noise tolerance across the 20 parameter initializations.

For the analyses in Figure 2.6, target trajectories were derived from neural recordings (M1, and SMA) during the four-cycle movements for each of the four condition types (forward-bottom-start, forward-top-start, backward-bottom-start, backward-top-start). Target trajectories spanned the time period from movement onset until 250 ms after movement offset. To emphasize that the network should complete the trajectory and remain in the final state, we extended the final sample of the target trajectory for an additional 500 ms. To obtain target trajectories, neural data were mean-centered and projected onto the top six PCs (computed for that condition). Each target trajectory was normalized by its greatest norm (across times). We trained a total of 160 networks, each with a different weight initialization. The eighty networks for each monkey included ten each for the two cortical areas and four condition types (two starting positions by two cycling directions).

Network dynamics were governed by equation 2.4, where $f := \tanh$ and $w \sim \mathcal{N}(0, \sigma_w^2 I)$ adds noise. v can be thought of as the membrane voltage and $f(v(t))$ as the firing rate. $Af(v(t))$ is then the vector of inputs to each unit: i.e., the firing rates weighted by the connection strengths. Network training attempted to minimize the difference between this input vector and a target trajectory: $s_{\text{targ}}(t)$. Training focused on the vector of inputs, rather than the vector of outputs (firing rates) purely for technical purposes. The end result is much the same as inputs and outputs are related by a monotonic function. A was trained using recursive least squares. The target trajectory was constructed as $s_{\text{targ}}(t) = Gy_{\text{targ}}(t).y_{\text{targ}}$ is the six-dimensional trajectory derived from the physiological data. G is an $N \times 6$ matrix of random weights, sampled from $\mathcal{U}\,[-.5, .5]$, that maps the global target trajectory onto a target input of each model unit. This construction ensures that the target network trajectory is isomorphic with the physiological trajectory, with each unit having random "tuning" for the underlying factors. The entries of A were initialized by draws from a centered normal distribution with variance $\frac{1}{N}$ (where N = 50, the number of network units). Simulation employed 4 ms time steps.

To begin a given training epoch, the initial state was set with $v(0)$ based on $s_{\text{targ}}(0)$ and A. The network was simulated, applying recursive least squares (Sussillo and Abbott, 2009) with parameter $\alpha = 1$ to modify A as time unfolds. After 1000 training epochs, stability was assessed by simulating the network 100 times, and computing the mean squared difference between the actual and target trajectory. That error was normalized by the variance of the target trajectory, yielding an R^2 value. An average (across the 100 simulated trials) $R^2 < 0.9$ was considered a failure.

Because the empirical population trajectories never perfectly repeated, it was trivially true that networks could follow the full trajectory, for both M1 and SMA, in the complete absence of noise (i.e., for σ_w=0). For the larger value of σ_w used for our primary analysis, all networks failed to follow the M1 trajectories while most networks successfully followed the SMA trajectories (although there were still some network initializations that never resulted in good solutions). It is of course unclear what value of σ_w is physiologically relevant. We therefore also performed an analysis where we swept the value of σ_w until failure. The level of noise that was tolerated was much greater when networks followed the SMA trajectories. Indeed, some M1 trajectories (for particular conditions) could never be consistently followed even at the lowest noise level tested.

To visualize network activity (Figure 2.6B-D) we "decoded" the network population. To do so, we reconstructed the first three dimensions of the trajectory (which should match the first three dimensions of the target trajectory) by pseudo-inverting G.

2.4.2 Predicting neural population activity

The optimization described by Equation 2.2 was performed using the Theano Python module. Optimization was initialized either with $\hat{X}_{\text{init}} = Z$, or with $\hat{X}_{\text{init}} = Z$ + noise where the noise was smooth with time but independent for each dimension. Both $\hat{X}$ and Z were $10 \times T$; they contained the projection onto the top ten PCs. T is the total number of timepoints across the conditions being considered. Specifically, we predicted neural activity for three middle cycles of forward cycling and three middle cycles of backward cycling (both taken from seven-cycle movements). Because dimensionality is equal for $\hat{X}$ and Z, the ability to decode Z from $\hat{X}$ will suffer as optimization modifies $\hat{X}$. However, because some dimensions of Z contain more variance than others, $\hat{X}$ can gain considerable new structure while compromising the decode only modestly. This tradeoff can be determined by the choice of λ. However, for scientific reasons, we employed a modified approach to better control that tradeoff. We wished to ensure that the predictions made by different cost functions all encoded muscle activity equally well. This aids interpretation when comparing the results of the optimization in Figure 2.7C,D with optimizations using different cost functions in Figure 2.S7. By matching encoding accuracy, any differences in similarity must be due to other structure that differs due to the cost function being optimized. Thus, instead of minimizing the first term of Equation 2.2 (which attempts to create a perfect decode) we minimized the squared difference between the decode R^2 and 0.95. We only considered optimizations that achieved this with a tolerance of 0.01. This approach insures that muscle encoding is equally good for the predicted populations responses yielded by different cost functions. Optimizations employed gradient descent using an inexact line search for the Wolfe conditions c_1=0.05 and c_2=0.1. As a technical point, the derivative used to compute $Q(t_{\text{end}})$ was based on the assumption that the three-cycle pattern would repeat.

2.4.2.1 Cost functions

Cost functions All cost functions were of the form:

$$\hat{X} = \underset{X}{\text{argmin}} \sum_{k=1}^{K} \lambda_k f_k(X, Z) \tag{2.6}$$

where f_k is some function of the input data and λ_k are scaling coefficients used to ensure that one term of the cost function did not dominate at the expense of the others. The arguments of $f_k()$ are the optimization variable, X and the empirical muscle activity, Z. All cost functions examined in Supplementary Figure 7 are described below in terms of different definitions of $f_k()$.

Muscle encoding and low tangling (same as Equation 2.2)

$$f_1(X, Z) = f_{\text{decode}}(X, Z) = ||Z - ZX^+X||_F^2 \tag{2.7}$$

$$f_2(X) = f_{\text{tangling}}(X) = \sum_t Q_X(t) \tag{2.8}$$

Nonlinear mapping with L-2 minimization

$$f_1(X, \bar{Z}) = f_{\text{decode-nonlin}}(X, \bar{Z}) = ||\bar{Z} - \hat{Z}||_F^2 \tag{2.9}$$

$\bar{Z}$ containsindividual muscle activity. Here we consider the activity of all muscles individually (rather than the top ten PCs as above) because this matters in the non-linear case. The hypothesis being considered is that motor cortex may use a simplified set of muscle "synergies" that becomes, via a set of non-linear transformations, the activity of each muscle. $Z = \boldsymbol{\alpha} + \tanh(\beta X + \boldsymbol{\gamma})$ with the parameters $\boldsymbol{\alpha}$, β, and $\boldsymbol{\gamma}$ optimized to minimize $f_{\text{decode-nonlin}}(X, \bar{Z})$.

$$f_2(X) = f_{\text{norm}}(X) = ||X||_F^2 \tag{2.10}$$

where F denotes the Frobenius norm.

Nonlinear mapping with tangling minimization:

$$f_1(X, \bar{Z}) = f_{\text{decode-nonlin}}(X, \bar{Z}) \tag{2.11}$$

$$f_2(X) = f_{\text{tangling}}(X) \tag{2.12}$$

where $f_{\text{decode-nonlin}}$ and f_{tangling} are as described above.

Low curvature:

$$f_1(X, Z) = f_{\text{decode}}(X, Z) \tag{2.13}$$

$$f_2(X) = f_{\text{curvature}}(X) = \sum_t \frac{||\dot{\mathbf{x}}_t^{\text{norm}} - \dot{\mathbf{x}}_{t-1}^{\text{norm}}||}{s_t} \tag{2.14}$$

where,

$$\dot{\mathbf{x}}_t^{\text{norm}} = \frac{\dot{\mathbf{x}}_t}{||\dot{\mathbf{x}}_t||} \tag{2.15}$$

and s_t is the normalized “speed" of the neural trajectory,

$$s_t = \frac{||\dot{\mathbf{x}}_t||}{\sum_{t'} ||\dot{\mathbf{x}}_{t'}||}. \tag{2.16}$$

As a technical point, we wished to ensure that the predictions made by different cost functions all encoded muscle activity equally well. By matching the accuracy of muscle encoding, any differences in similarity must be due to other structure introduced during optimization. We therefore modified $f_{\text{decode}}(X, Z)$ and $f_{\text{decode-nonlin}}(X, \bar{Z})$ so that they were minimized when decode accuracy had an R^2 of 0.95, rather than 1.0. We only considered optimizations that achieved this with a tolerance of 0.01.

2.4.3 Similarity between empirical and predicted data

We assessed similarity using a modified version of canonical correlation [123]. This method finds a pair of orthogonal transformations, one for each dataset, that maximizes the correlation between the transformed datasets. Specifically, for mean-centered datasets $X_a \in \mathcal{R}^{K\times T}$ and $X_b \in \mathcal{R}^{K\times T}$, similarity is:

$$S(X_a, X_b) = \underset{M_a, M_b}{\text{argmax}} \frac{tr(M_a^\top X_a X_b^\top M_b)}{\sqrt{tr(M_a^\top X_a X_a^\top M_a) tr(M_b^\top X_b X_b^\top M_b)}} \tag{2.17}$$

Subject to the constraint that M_a and M_b are orthonormal matrices. Similarity will thus be unity if two datasets are the same but for an orthonormal transformation. Note also that an overall shift of one dataset relative to the other does not impact similarity because the data are mean-centered before computing similarity. Due to the normalization in the denominator of the above cost function, similarity is also not impacted by an isotropic scaling of one dataset relative to the other.

Chapter 3: Approximating exponential family models (not single distributions) with a two-network architecture

In the remainder of this dissertation, we turn our focus to the development and application of deep generative modeling techniques for theoretical neuroscience. Oftentimes, we seek a Bayesian posterior given some choices of likelihood and prior, which make the posterior analytically unavailable. Variational inference, an optimization technique for fitting posteriors, is generally used to approximate singular posterior distributions for a chosen dataset. However, we may want to run the same approximate posterior inference program repeatedly (for the same likelihood and prior) for many similar datasets. Here, we introduce an algorithm to train two-network architectures to approximate exponential family models (not just single distributions).

The natural parameter of the exponential family model is input to the first neural network, which determines the weights and biases of the normalizing flow. The normalizing flow is used to approximate the exponential family distribution indexed by the chosen natural parameter. By applying this technique to intractable exponential family posteriors, we enable look-up posterior inference for arbitrary combinations of dataset and prior.

We demonstrate the efficiency of this technique with a log-gaussian poisson model of primary visual cortex spike responses to drift gratings. The log-gaussian poisson model of neural spikes is a statistical generative model, which can be used to evaluate descriptive theories of neural circuits or support the development of mechanistic hypotheses (see types of theories in Section 1.1). By using exponential family networks, neuroscientists can save time and computational resources when inferring firing rates (or other parameters) in statistical generative models belonging to the exponential family.

The remainder of this chapter was co-authored by John Cunningham [124].

3.1 Introduction

Much recent work has focused on deep generative models, which map a latent random variable $w \sim q_0$ through a member of a highly expressive function family $\mathcal{G} = \{g_\theta : \theta \in \Theta\}$, the composition resulting in an implicit probability model $\mathcal{M} = \{q(g_\theta(w)) : \theta \in \Theta\}$. Choosing $\mathcal{G}$ to be a parameter-indexed family of neural networks has both a rich history [125, 126], and has recently been used to produce exciting results for density estimation [127, 128, 71], generation of complex data [129], variational inference [130, 33, 131], and more.

On the other hand, since these models have been chosen to be generic and flexible, they can lack the classic stipulation that a model instantiates existing domain knowledge [132, 133, 134]. There are well known drawbacks of fitting such flexible models to finite (albeit large) data sets, which contrast with the bias-variance benefits that come from working in a restricted model space [135, §7.3]. Work on generalization and compressibility in deep networks suggests that this broad class of function families are indeed quite large, perhaps problematically so [136].

When performing inference on a restricted model, it is increasingly common to deploy an implicit "recognition network" model for variational inference [130], which finds a $q_{\theta^*}(z) \in \mathcal{M}$ such that an evidence bound is optimized with respect to the true posterior $p(z|X)$. However, it is widely understood that many such true posteriors $p(z|X)$ are exponential families (albeit intractable, due to the choice of sufficient statistics $t(z)$) of the form: $\mathcal{P} = \left\{\frac{h(z)}{A(\eta)} \exp\{\eta^\top t(z)\} : \eta \in H\right\}$ [137]. Should we be able to learn a tractable approximation to this exponential family model, we would in the very least get the bias-variance benefits of an intelligently restricted model space, and at best would get inference "for free" in the sense that we could evaluate approximate posteriors directly without separate optimization for each dataset encountered (a novel form of *amortized inference* [138, 130, 33, 139]). In this paper, we aim to learn a restricted model $\mathcal{Q} = \{q(z;\eta) : \eta \in H\}$ that will be a strict subset of $\mathcal{M}$ and will closely approximate a target exponential family $\mathcal{P}$. Note the critical difference between this aim and much of the literature that seeks to learn a density $q_\theta^* \in \mathcal{M}$ (we explore this distinction in depth both algorithmically and empirically). To proceed, we specify a set of invertible deep generative models $\mathbb{Q} = \left\{Q_\phi : \phi \in \Phi\right\}$, from which we can learn a single model Q_{ϕ^*}. We restrict Θ, the parameter space of $\mathcal{M}$, to be itself the image of a second deep *parameter network* family $\mathcal{F} = \left\{f_\phi : \phi \in \Phi\right\}$, such that $\left\{f_\phi(\eta) : \eta \in H\right\} \subset \Theta$.

We define this two-network architecture, which we term an *exponential family network* (EFN), and we specify a stochastic optimization procedure over an ELBO-like variant of the typical Kullback-Leibler divergence. We then demonstrate the ability of EFNs to approximately learn exponential families and the benefits of approximating distributions in such restricted model spaces. Finally we demonstrate the computational savings afforded by this approach when learning the posterior family of point-process latent intensities, given neural spike trains recorded in a neuroscience experiment.

3.2 Exponential family networks

To define exponential family networks (EFNs), we begin with relevant context for our modeling choice of exponential families (§2.1). We then describe the network architectural constraint and the background we use to satisfy that constraint (§2.2). We then introduce EFN in detail, including the optimization algorithm used for learning (§2.3). The similarities with variational inference are then explored in depth in (§2.4).

3.2.1 Exponential families as target model $\mathcal{P}$

We will focus on a fundamental problem setup in probabilistic inference, that of a latent variable $z \in \mathcal{Z}$ with prior belief $p_0(z)$, and where we observe a dataset $X = \{x_1, ..., x_N\} \subset \mathcal{X}$ as conditionally independent draws given z. Updating our belief with data produces the posterior $p(z|X) \propto p_0(z) \prod_{i=1}^N p(x_i|z)$. This setup is shown as a graphical model in Figure 1A.

If we restrict our attention to priors and likelihoods that belong to exponential families $\mathcal{P} = \left\{\frac{h(\cdot)}{A(\eta)} \exp\{\eta^\top t(\cdot)\} : \eta \in H\right\}$, the posterior can also be viewed as an exponential family, albeit an intractable one [137]. For simplicity we will hereafter suppress the base measure $h(\cdot)$. Consider:

$$p_0(z) = \frac{1}{A_0(\alpha)} \exp\left\{\alpha^\top t_0(z)\right\} \tag{3.1}$$

$$p(x_i|z) = \frac{1}{A(z)} \exp\left\{\nu(z)^\top t(x_i)\right\}, \tag{3.2}$$

where $t(\cdot)$ is the sufficient statistic vector, and $\nu(z)$ is the natural parameter of the likelihood in natural form [140]. The posterior then has the form:

$$p(z|x_1, ..., x_N) \propto \exp\left\{\begin{bmatrix} \alpha \\ \sum_i t(x_i) \\ -N \end{bmatrix}^\top \begin{bmatrix} t_0(z) \\ \nu(z) \\ \log A(z) \end{bmatrix}\right\}, \tag{3.3}$$

which again is an intractable exponential family.

To give a concrete example, consider the hierarchical Dirichlet – a Dirichlet prior $z \sim Dir(\alpha)$ (of dimension $|\mathcal{Z}|$) with conditionally iid Dirichlet draws $x_i|z \sim Dir(\beta z)$ [141, 142, 143, 144]). Figure 1B shows the prior for a given α (top), and three examples of datasets that could arise via this generative model (middle). A set of basic manipulations shows the hierarchical Dirichlet posterior $p(z|X)$ to be itself an exponential family with natural parameter $\eta = \left[\alpha - 1, \sum_i \log(x_i), -N\right]^\top$ and sufficient statistic $t(z) = \left[\log(z), \beta z, \log(B(\beta z))\right]^\top$. The corresponding posteriors are shown in Figure 1B (bottom). Note importantly that, because the likelihood was chosen to be an exponential family (which is closed under sampling), this form will not change for any choice of $|Z|$-dimensional hierarchical Dirichlet – any draw from the prior, any N, or any particular realization of observed data X (technically the prior need not be exponential family, but we leave it as such for simplicity). The exponential family is clearly sufficient for this property, and the Pitman-Koopman Lemma further clarifies that it is also necessary (under reasonable conditions) [140, §3.3.3].

The critical observation here is that, if we can approximately learn an intractable exponential family (the model itself), then it becomes trivial to perform posterior inference. To execute posterior inference, we simply construct the natural parameter η by concatenating the hyperparameters of the prior, the summed sufficient statistics over the dataset, and the total number of samples in the dataset (see Equation 3.3). Following the minor amount of computation required to construct the natural parameter η, it is fed into the parameter network of the EFN, and the posterior distribution is then produced by the density network (see next section). The goal of EFNs is to amortized inference across the posterior family $\mathcal{P}$ for many choices of η, which is determined by hyperparameterization of the prior and individual datasets of varying sample count. By fitting EFN's, we can save computation when we need to do inference in many instances of the same posterior family model (e.g. upon many datasets).

3.2.2 Density networks as approximating family $\mathcal{M}$

Invertible deep generative models, which we will use for our approximating model family $\mathcal{M}$, can be defined by any base random variable $w \sim p_0$ mapped through any bijective, parameter-indexed function family $\mathcal{G} = \{g_\theta : \theta \in \Theta\}$, with induced density on $z = g_\theta(w)$ as $q_\theta(z)$. This is a well-established idea that has recently seen many variants and applications [126, 145, 146, 128, 127, 68, 147, 71, 148]. Specifically, let $z = g_\theta(w) = g_L \circ ... \circ g_1(w)$ for bijective vector-valued functions g_ℓ (surpressing θ), and denote $J_\theta^\ell(z)$ as the Jacobian of the function g_ℓ at the layer activation corresponding to z. Then we have:

$$q_\theta(z) = q_0\left(g_1^{-1} \circ ... \circ g_L^{-1}(z)\right) \prod_{\ell=1}^{L} \frac{1}{|J_\theta^\ell(z)|}. \tag{3.4}$$

The specific form of the layers g_ℓ can be chosen based on empirical considerations; we used planar flow architectures [68]. For the remainder (and to avoid confusion when we introduce a second network), we call this deep bijective neural architecture the *density network*; this network is shown vertically oriented (flowing from w down to z) in Figure 1C.

This density network induces the model $\mathcal{M} = \{q(g_\theta(w)) : \theta \in \Theta\}$, which previous work has searched to find a single optimized distribution q_{θ^*} (such as a posterior or data generative density), on the assumption and subsequent empirical evidence that the target exponential family member is close to (or approximately belongs to) $\mathcal{M}$. We make the same assumption for the exponential family itself and seek to intelligently restrict $\mathcal{M}$ in order to learn the exponential family.

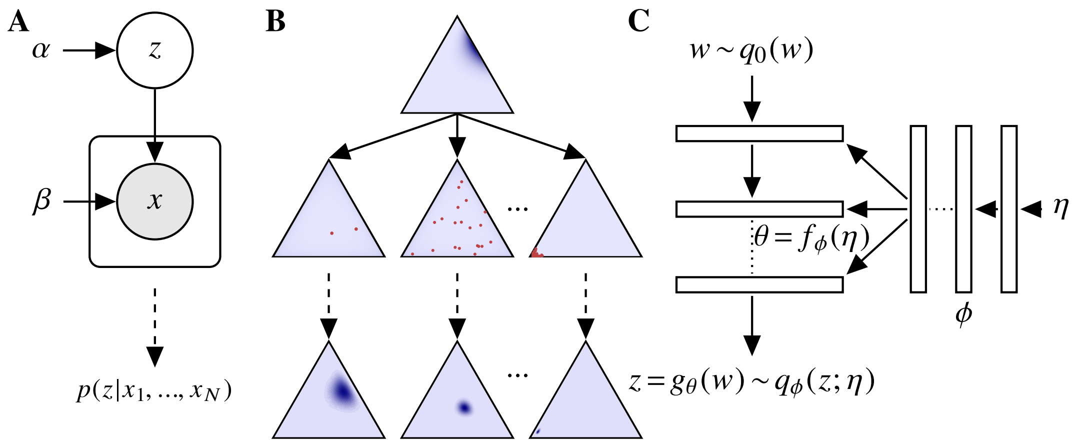


Figure 3.1: (A) Probabilistic graphical model. (B) Hierarchical Dirichlets: a Dirichlet prior with conditionally iid Dirichlet draws. (top) prior $p_0(z)$, (middle) three sample conditional Dirichlet datasets X of $N = 2$, 20, and 100, and (bottom) the three posteriors that themselves belong to an exponential family $\mathcal{P}$. (C) Architecture for exponential family network (EFN): density network running top to bottom; parameter network right to left.

3.2.3 Exponential family networks as approximating model Q

Having introduced our target model $\mathcal{P}$, an exponential family with natural parameters $\eta \in H$, and the density network family $\mathcal{M}$, we now seek to learn $Q \approx \mathcal{P}$, where $Q \subset \mathcal{M}$. To do so we will parameterize θ, the parameters of the density network, as the image of a second *parameter network* family $\mathcal{F} = \left\{ f_\phi : H \to \Theta, \phi \in \Phi \right\}$. This network is shown flowing from right to left in Figure 1C. Using a second meta-network to aid or restrict network learning has been used in a variety of settings; a few examples include parameterizing the optimization algorithm in the "learning to learn" setting [149], and a more closely related work that used a second network to condition on observations for local latent variational inference [68], a connection which we explore closely in Appendix A.

Any choice of parameter network parameters ϕ induces a $|H|$-dimensional submanifold (the image $f_\phi(H)$) of the density network parameter space Θ, and as such defines a restricted model $Q_\phi = \left\{ q_{f_\phi}(z;\eta) : \eta \in H \right\} \subset \mathcal{M}$; by our choice of H as the natural parameter space of the exponential family target $\mathcal{P}$, this model restriction is at least of the correct dimensionality. Our goal then is to search over the implied set of models $\mathbb{Q} = \left\{ Q_\phi : \phi \in \Phi \right\}$ to find an optimal ϕ^* such that

$\mathcal{Q}_{\phi^*} \approx \mathcal{P}$.

Given the connections between the exponential family and Shannon entropy, we will measure the error between $\mathcal{Q}_\phi$ and $\mathcal{P}$ with Kullback-Leibler divergence. Consider for the moment a fixed choice of natural parameter η; we seek to minimize, over ϕ:

$$D\left(q_\phi(z;\eta)||p(z;\eta)\right) = \mathbb{E}_{q_\phi}\left(\log q_\phi(z;\eta) - \eta^\top t(z) + \log(A(z))\right) \tag{3.5}$$

which is equivalent to minimizing

$$\mathbb{E}_{q_\phi}\left(\log q_\phi(z;\eta) - \eta^\top t(z)\right) = \mathbb{E}_{q_\phi}\left(\log q_0\left(g_\theta^{-1}(z)\right) + \sum_{\ell=1}^{L} \log |J_\theta^\ell(z)| - \eta^\top t(z))\right), \tag{3.6}$$

where again we note that $\theta = f_\phi(\eta)$, and thus for a fixed η, this objective depends only on ϕ. Indeed, the target $\eta^\top t(z)$ is linear in η (an obvious restatement of the log-linear exponential family form), giving us some hope that we may be able to learn this model [1] .

Of course we seek to approximate not just a single target exponential family member ($p(z;\eta)$ for a fixed η), but rather the entire model $\mathcal{P} = \{p(z;\eta) : \eta \in H\}$. For optimization we thus need to introduce a distribution $p(\eta)$ (for stochastic optimization), leading to the objective:

$$\underset{\phi}{\operatorname{argmin}}\, \mathbb{E}_{p(\eta)}\left(D\left(q_\phi(z;\eta)||p(z;\eta)\right)\right) = \underset{\phi}{\operatorname{argmin}}\, D\left(q_\phi(z;\eta)p(\eta)||p(z;\eta)p(\eta)\right). \tag{3.7}$$

Unbiased estimates of this objective are immediate. $q_\phi(z;\eta)$ is sampled by computing the density network parameters $\theta = f_\phi(\eta)$ (using the parameter network), sampling the latent $w \sim q_0(w)$, and running that w through the density network. $p(\eta)$ is user defined and chosen such that it is trivial to sample. Stochastic optimization can then be carried out on the estimator:

$$\mathbb{L}(\phi) = \frac{1}{K}\frac{1}{M}\sum_{k=1}^{K}\sum_{m=1}^{M}\left(\log q_0\left(g_{\theta^k}^{-1}\left(z^m\right)\right) + \sum_{\ell=1}^{L}\log|J_{\theta^k}^\ell\left(z^m\right)| - \eta_k^\top t\left(z^m\right)\right), \tag{3.8}$$

[1]This objective can also produce approximations of the log partition (as the intercept term implied by this linear target), which we have found to be reasonably accurate, though nuanced schemes are likely appropriate [150].

where $\theta^k = f_\phi(\eta_k)$. Successful optimization over ϕ should thus result in $\mathcal{Q}_{\phi^*} \in \mathcal{M}$ that accurately approximates the target exponential family; that is, $\mathcal{Q} \approx \mathcal{P}$. We call this two-network architecture and optimization an exponential family network (EFN). What remains for empirical implementation is to make particular choices of hyperparameters, network layers, and optimization algorithm, which we specify in §3 below.

3.2.4 Relation to variational inference

A tremendous amount of work in recent years has gone into variational inference (VI), and its similarity to EFN warrants careful attention. In the following, we aim to carefully (and somewhat pedantically) dissect this question. As such, though EFN can address any target exponential familiy, to bring us closest to VI let us here restrict the EFN target model $\mathcal{P}$ to be a family of posterior distributions (such as for example the log-Gaussian Poisson example in Section 4.2.).

The typical role of variational inference is to infer an approximate posterior $q_\phi(z) \approx p(z|X)$. In this setting, the difference with EFN is stark, in so much as VI learns this single posterior approximation, whereas the main goal of the EFN is to approximate the model $\mathcal{P} = p_\eta(z|X) : \eta \in H$: to learn the family of distributions. More recently, much focus has gone into the particular instance of VI for local variables z_i, for example $\prod_{i=1}^N p(z_i)p(x_i|z_i)$ (such as a variational autoencoder [130]) or $p(u)\prod_{i=1}^N p(z_i|u)p(x_i|z_i)$ (latent Dirichlet allocation being a canonical example [143, 151]), the result of which is often an amortized inference/recognition network that produces a local variational distribution $q_{\phi^*}(z_i|x_i)$. This local variational distribution is typically parameterized explicitly: the inference network $\mu_\phi(x_i)$ induces a local parametric distribution, often a Gaussian $q(z_i|x_i) \sim \mathcal{N}\left(z_i; \mu_\phi(x_i)\right)$ [130, for example]. Viewed this way, local-latent-variable VI methods induce a model $\left\{q_{\phi^*}(z_i|x_i) : x_i \in X\right\}$ for a finite dataset X. In that sense, EFN and VI are similar 'model learning' approaches. Even more closely, as part of a long-standing desire to add structure to VI beyond mean-field (classically [152, 153]; more recently [154, 155], to name but a few), in several cases an inference network has been used to parameterize a deep implicit model (in a two-network inference architecture, to say nothing of whether or not the generative model itself is a deep generative model); closest to the EFN architecture is [68] (cf. Figure 2 of [68] with Figure 1C here). Thus EFN (when used for posterior families) can be seen as a close generalization of VI.

Even accepting this VI-as-a-model view, the difference between the finite dataset X and the natural parameter space H persists when viewed at a mechanical level; well-known are the overfitting/generalization issues associated with a finite dataset compared with access to a distribution $p(\eta)$. Thus one goal of EFN is to allow the model $Q_{\phi^*} \approx \mathcal{P}$ to be learned in the absence of a finite dataset, such that inference on that dataset can then be executed without concerns of overfitting to that set (and of course without having to run a VI optimization for every new dataset; we will demonstrate this benefit of EFN in the experiments). Perhaps more importantly, the "model" implied by VI is parameterized by x_i, and indeed the inference network takes x_i as input. The EFN on the other hand is considerably more general; the posterior includes the natural parameters of the prior (Equation 3.3). This allows the EFN architecture to learn across a more general setting that VI cannot, since any VI inference network is only parameterized by data. One final difference made clear by Equation 3.3 is that the observations are given to the EFN *in natural form* (that is, $t(x_i)$, not x_i) [140]. This choice is a novel insight: by exploiting the known sufficiency of $t(x_i)$ in the target model $\mathcal{P}$, some difference in performance for VI may be observed. Accordingly, while EFN and VI do at a high level bear multiple similarities, the differences are both material and provoke interesting speculation about means to improve both VI and EFN.

3.3 Results

To investigate the performance of EFNs, we assess approximation fidelity on some tractable exponential families, examine the benefits of learning in a regularized model space, and characterize data analysis scenarios in which training an EFN is computationally advantageous. First, we test the ability of EFNs to approximate the target model $\mathcal{P}$ when this model is a known, tractable exponential family: this choice provides a simple ground truth and calibrates us to expected performance vs alternatives. Additionally, tractable exponential families allow us to measure the relative accuracy of single distribution approximations in isolation versus indexed members of trained EFNs. The main advantage of learning an EFN is to make tractable a previously intractable exponential family (at least approximately). This confers major benefits in terms of test-time: for example, rather than optimization needing to be run for variational inference with each particular dataset realized from a model class, EFN will allow immediate lookup. This benefit is orders of magnitude and is not instructive to view, so we show a decision boundary among neural data analysis scenarios, in which training an EFN is computationally advantageous to approximating several distributions through VI optimization individually. Most often, training an EFN has striking computational advantages.

To compare model approximations by EFNs to standard methodology, we alternatively train density networks to approximate members of the target model family. Since η will not change, we dispose of the parameter network and train the density network directly over θ (again with a deterministic choice of a single η). When the distribution being approximated is a posterior, this procedure is variational inference. This is the key comparison for the EFN model, and we refer to this alternative as NF for normalizing flow.

We also must make some particular architectural choices for these experiments. We considered a variety of density network architectures. For each exponential family, we searched through some candidate architectures which consisted of cascades of normalizing flow layers such as planar and radial flows introduced in [68], a structured spinner flows inspired by [156], and a single affine

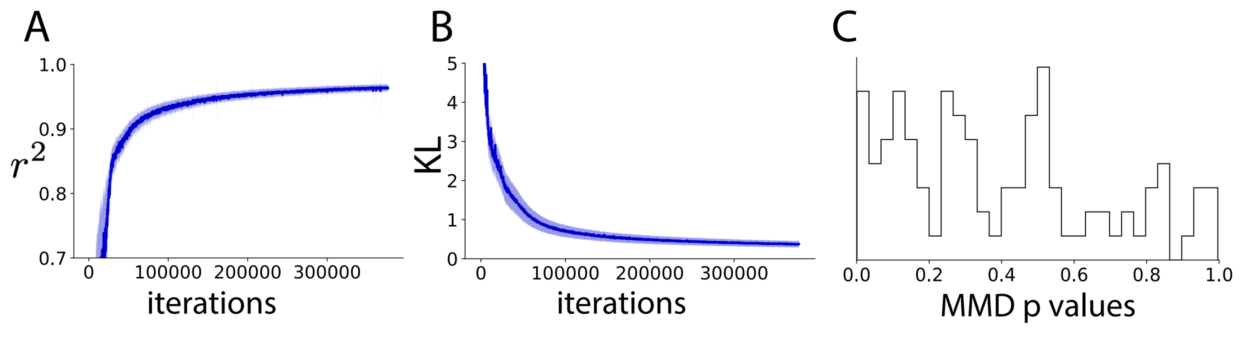


Figure 3.2: 50-dimensional Dirichlet exponential family network. (A) Distribution of r^2 between log density of EFN samples and ground truth across choices of η throughout optimization. (B) Distribution of KL divergence throughout optimization. (C) Distribution of maximum mean discrepancy p-values between EFN samples and ground truth after optimization.

transformation.

The parameter network was given tanh nonlinearities. In many of the results below we will analyze EFNs across a range of model dimensionality D (that is, $z \in \mathcal{Z} \subseteq \mathbb{R}^D$). In all cases then we have also D flow layers in the density network (except when the affine transformation is optimal). In analyses where D was less than 20, 20 flow layers were used. The number of layers in the parameter network scaled as the square root of D, with a minimum of 4 layers, and the number of units per layer scaled linearly from the input to the number of density network parameters. Models were trained using the Adam optimizer algorithm [157], with learning rates ranging from 10^{-3} to 10^{-5}. Optimizations ran for at least 50,000 iterations, and completed once there was a subthreshold increase in ELBO. These choices were made so that model performance saturated, and were held constant within comparative analyses.

All code was implemented in tensorflow, and is available at
`https://github.com/cunningham-lab/efn`.

3.3.1 Tractable exponential families

Here we study the multivariate Gaussian and Dirichlet families, which offer a known ground truth and intuition about the range of performance that EFN – learning a model – has with respect to its single-distribution counterpart NF. While this section serves primarily to validate the approach of EFN, such approximations to popular exponential family models may serve as differentiable generative modules in hierarchical generative models.

First, to validate the basic EFN approach, we train the $D = 50$-dimensional Dirichlet family. We chose $p(\eta)$, the prior on the α parameter vector of the Dirichlet, as $\alpha_i \sim U[.5, 5.0]$. The number of η samples K at each iteration was 100, and the minibatch size in z was $M = 1000$. Figure 2 shows a high accuracy fit to this Dirichlet model: Figures 2A and 2B shows rapid convergence to high coefficient of determination r^2 and low Kullback-Leibler divergence. Since we are doing distribution regression, r^2 is a convenient metric calculated as the coefficient of determination between the model predictions $\log(q_\phi(z_i; \eta_k))$ and their known targets $\eta_k^\top t(z_i)$. We can then perform

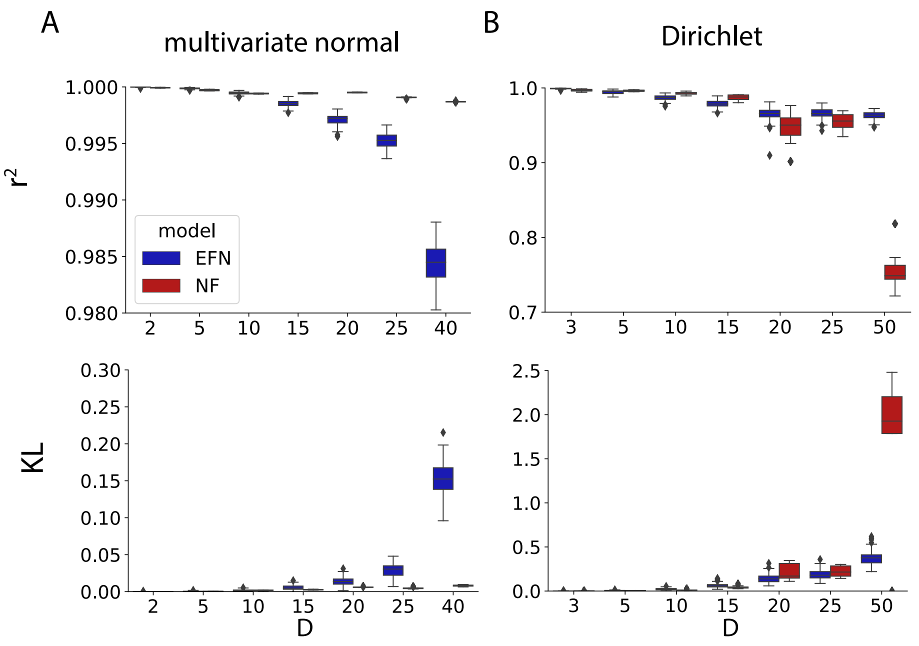


Figure 3.3: Scaling exponential family networks: D denotes the dimensionality of the family being learned, and comparisons are between EFN and its alternative NF (see text). (A) Multivariate normal family (B) Dirichlet family.

a standard MMD-based kernel two-sample test [158] between distributions chosen from $\mathcal{P}$ and Q_{ϕ^*}. Since an appreciable majority of exponential family distributions of the EFN model Q_{ϕ^*} are not significantly different from the ground truth distribution of the true target Dirichlet family $\mathcal{P}$ (100-sample tests), we consider the Dirichlet model to be well-approximated by the EFN.

Second, in Figure 3 we consider how this performance scales across dimensionality. Consider EFN vs NF, where again the only difference is that EFN attempts to learn the entire model (as in $\eta \in H$), whereas NF chooses a single η and thus learns a single distribution optimizing the density network parameters θ directly. One might expect a noticeable deficit in approximation by EFNs, since they are generalizing the expressivity of the density network across $p(\eta)$. Accordingly, this deficit is apparent when modeling the multivariate normal family (Fig. 3A). In low dimensions, we have nearly exact model approximation by EFNs (blue) and distributional approximations by NFs (red). The distributions learned by NFs were drawn from the same η prior as the EFN was trained. However, as dimensionality increases EFN distributional approximations become significantly worse than the nearly perfect approximations learned by NFs. The η prior of the multivariate normal was specified as an isotropic normal on the mean parameter $\mu_i \sim \mathcal{N}(0, 0.1)$, and an inverse-Wishart distribution on the covariance $\Sigma \sim IW(n, \Psi)$ with degrees of freedom $n = 5$ and $\Psi = nDI$.

However, learning the model with EFN does not necessarily harm the distributional approximation relative to NF. In fact, conventional wisdom suggests that learning in a restricted model space is beneficial for regularization. Here, the expansiveness of the η prior determines the necessary degree of generalization of the EFN assigning a weight in the objective to the approximation loss of each distribution. By requiring the parameter network to learn generalizations of the density network across the η prior, local minima may be avoided that NFs would otherwise be susceptible to. This is in fact what we see when modeling the Dirichlet distribution (Fig. 3B). In low dimensions, NF performs better than EFN, but from 20 dimensions and greater, the restricted model space of the EFN confers superior optimization convergence relative to NF, which is more susceptible to local minima.

3.3.2 Lookup inference in an intractable exponential family

Of course the main interest of an EFN is to learn intractable exponential families. The Gaussian family is the ubiquitous prior for real valued parameters, but it does not match well with the non-negativity requirements of the intensity measure required of certain distributions, most notably the Poisson. Log Gaussian Cox Processes have been used numerous times in machine learning, and all have required attention to approximate inference in this fundamentally nonconjugate model. Furthermore, many of these examples have been used to analyze the latent firing intensity of neural spike train data [159, 160, 161, 34].

We demonstrate the utility of a log-Gaussian Poisson EFN for inferring latent firing intensities of neurons recorded in primary visual cortex of anesthetized macaques in response to 6.25 Hz drift grating stimuli [162]. 200 spike train responses were recorded for each neuron in response to 12 different grating orientations. Spiking responses were binned into 20ms intervals from 280ms-680ms following stimulus onset (to avoid the effects of transient neural dynamics). The latent space of this model was thus 20-dimensional and represents the log-firing rates of single neurons

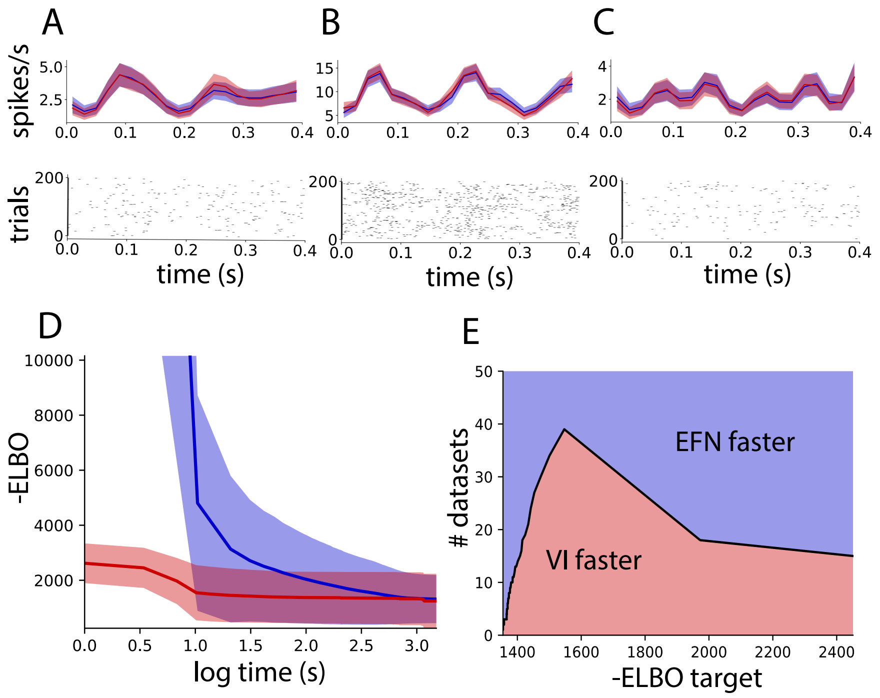


Figure 3.4: Lookup inference in a log-Gaussian Poisson model with V1 responses to drift grating stimuli. (A-C) Top: Inferred latent intensities from a single EFN (blue) or varitional inference (red) run individually for each dataset. Shading denotes the standard deviation of the posterior. Bottom: Corresponding V1 spiking responses. (D) Distribution of -ELBO throughout training across a held out test group of 100 datasets for the EFN (blue), and across 298 datasets fit with NF (red). (E) Decision boundary for what number of datasets for a given target approximation accuracy it is advantageous to train an EFN rather than run variational inference individually for each dataset.

in trial-averaged responses to particular stimulus conditions. The frequency of the drift grating stimulus motivated a multivariate gaussian prior corresponding to a gaussian process with a 25ms squared-exponential kernel. The mean and variance calculated across log firing rates of all neural responses (or "datasets") determined the final hyperparameterization (μ and Σ) of the gaussian prior. Across three experimental subects, 247 neurons with signal-to-noise ratios greater than 1.5 and mean firing rates greater than 1 Hz were considered, resulting in 2,964 total datasets for inferring latent intensities. By training an EFN on this log-Gaussian Poisson family, we have a model of the posterior distribution for this prior covariance, and some chosen spiking responses.

We can compare the posterior distribution learned with standard variational inference with NF (red) for a given neuron's response, to the posterior distribution we get with immediate lookup by supplying the spiking responses of a neuron and the chosen prior (the natural parameters of the posterior) as input to a trained EFN (blue) (Fig. 4A-C). As a reminder, NF is learning a single member of an exponential family. If that exponential family is in fact an intractable posterior distribution (such as the hierarchical Dirichlet or log-Gaussian Poisson examples already discussed), then indeed NF is *precisely* performing variational inference with a normalizing flow recognition network, as in [68, 147, 71]. Both EFNs and NFs were trained with 30 planar flow layers. These posteriors are very similar, and neither appears to fit the data better than the other. The high quality of these lookup posteriors is an incredible feat by the EFN. Now that we have trained this EFN, we have immediate posterior inference for all remaining and future neural recordings.

Training an EFN understandably takes more time than an NF (Fig. 4D), but once the EFN is trained we have immediate posterior inference lookup. If we have a target level of approximation (ELBO target) we can determine when it is faster to get posterior inference on a number of datasets by training an EFN and then using the immediate lookup feature or by running variational inference independently for each distribution. By computing the amount of computational time it takes to reach the ELBO target on average for both EFN and NF, and then counting how many datasets it would take to learn with NF before eclipsing the training time for the EFN. This results in a decision boundary (Fig. 4E), where an EFN is more computationally efficient for running posterior inference, and we have infinite computational savings for each additional dataset. As the ELBO target increases from its minimum value on the right of Fig. 4E, the extra time it takes an EFN to reach this ELBO target relative to NF increases initially. At some point, the EFN ELBO and NF ELBO distributions begin to converge (Fig. 4D), and the gap of time between learning an EFN and an NF for a given ELBO target begins to decrease. For some posterior distributions, the EFN learning approach may confer a mean ELBO greater than achievable by traditional variational inference due to the benefits of learning in a restricted model class. In the case where EFN achieves a greater mean ELBO than NF, it is always advantageous to use EFN.

Our ability to approximate an intractable exponential family model with an EFN is very encouraging. We have shown in the applied setting of inferring neural firing rates that learning the posterior inference model with an EFN can confer enormous computational savings. There is nothing unique about this application, insofar as we expect the power of learning exponential family models to translate to applications of intractable exponential family models in other settings. One can imagine downloading a pre-trained EFN for an intractable exponential family model, and being able to do posterior inference immediately given an arbitrary choice of prior and dataset.

3.4 Discussion

We have approached the problem of learning an exponential family using a deep generative network, the parameters of which are the image of the natural parameters of the target exponential family under another deep neural network. We demonstrated high quality empirical performance across a range of dimensionalities, the potential for better approximations when learning in a restricted model space, and computational savings afforded by immediate posterior inference lookup.

Chapter 4: Emergent property inference captures complex parametric structure of neural circuit models and scales to high dimensions

In Chapter 3, we presented a deep generative modeling technique that uses normalizing flows for inference in statistical generative models, and demonstrated its efficiency on neural datasets. In this chapter, we introduce an additional deep generative modeling technique, which employs normalizing flows to infer distributions of parameters of neural circuit models that produce emergent properties of computation. Emergent property inference (EPI) is a machine learning technique designed to build and evaluate mechanistic theories of neural circuits. We compare EPI to alternative techniques for inference in neural circuit models, and in Section 4.5, we draw connections between these techniques and the exponential family networks of Chapter 3. The remaining content of this chapter (Sections 4.1-4.4.3) corresponds to lightly adapted sections of Bittner et al. 2021 [163] concerning the introduction and evaluation of EPI, and were co-authored by Agostina Palmigiano, Alex T. Piet, Chunyu A. Duan, Carlos D. Brody, Kenneth D. Miller, and John P. Cunningham.

4.1 Introduction

The fundamental practice of theoretical neuroscience is to use a mathematical model to understand neural computation, whether that computation enables perception, action, or some intermediate processing. A neural circuit is systematized with a set of equations – the model – and these equations are motivated by biophysics, neurophysiology, and other conceptual considerations [8, 9, 10, 11, 12]. The function of this system is governed by the choice of model *parameters*, which when configured in a particular way, give rise to a measurable signature of a computation. The work of analyzing a model then requires solving the inverse problem: given a computation of interest, how can we reason about the distribution of parameters that give rise to it? The inverse problem is crucial for reasoning about likely parameter values, uniquenesses and degeneracies, and predictions made by the model [164, 165, 166].

Ideally, one carefully designs a model and analytically derives how computational properties determine model parameters. Seminal examples of this gold standard include our field's understanding of memory capacity in associative neural networks [167], chaos and autocorrelation timescales in random neural networks [168], central pattern generation [169], the paradoxical effect [170], and decision making [171]. Unfortunately, as circuit models include more biological realism, theory via analytical derivation becomes intractable. Absent this analysis, statistical inference offers a toolkit by which to solve the inverse problem by identifying, at least approximately, the distribution of parameters that produce computations in a biologically realistic model [172, 173, 174, 175, 176, 177].

Statistical inference, of course, requires quantification of the sometimes vague term *computation*. In neuroscience, two perspectives are dominant. First, often we directly use an *exemplar dataset*: a collection of samples that express the computation of interest, this data being gathered either experimentally in the lab or from a computer simulation. Though a natural choice given its connection to experiment [17], some drawbacks exist: these data are well known to have features irrelevant to the computation of interest [178, 179, 180], confounding inferences made on such data. Related to this point, use of a conventional dataset encourages conventional data likelihoods or loss functions, which focus on some global metric like squared error or marginal evidence, rather than the computation itself.

Alternatively, researchers often quantify an *emergent property* (EP): a statistic of data that directly quantifies the computation of interest, wherein the dataset is implicit. While such a choice may seem esoteric, it is not: the above "gold standard" examples [167, 168, 169, 170, 171] all quantify and focus on some derived feature of the data, rather than the data drawn from the model. An emergent property is of course a dataset by another name, but it suggests different approach to solving the same inverse problem: here we directly specify the desired emergent property – a statistic of data drawn from the model – and the value we wish that property to have, and we set up an optimization program to find the distribution of parameters that produce this computation. This statistical framework is not new: it is intimately connected to the literature on approximate bayesian computation [51, 181, 52], parameter sensitivity analyses [182, 183, 184, 185], maximum entropy modeling [186, 187, 188], and approximate bayesian inference [189, 190]; we detail these connections in Section 4.4.1.1.

The parameter distributions producing a computation may be curved or multimodal along various parameter axes and combinations. It is by quantifying this complex structure that emergent property inference offers scientific insight. Traditional approximation families (e.g. mean-field or mixture of gaussians) are limited in the distributional structure they may learn. To address such restrictions on expressivity, advances in machine learning have used deep probability distributions as flexible approximating families for such complicated distributions [68, 69] (see Section 4.4.1.2). However, the adaptation of deep probability distributions to the problem of theoretical circuit analysis requires recent developments in deep learning for constrained optimization [191], and architectural choices for efficient and expressive deep generative modeling [72, 192]. We detail our method, which we call emergent property inference (EPI) in Section 4.2.2.

Equipped with this method, we demonstrate the capabilities of EPI and present novel theoretical findings from its analysis. First, we show EPI's ability to handle biologically realistic circuit models using a five-neuron model of the stomatogastric ganglion [41]: a neural circuit whose parametric degeneracy is closely studied [193]. Then, we show EPI's scalability to high dimensional parameter distributions by inferring connectivities of recurrent neural networks that exhibit stable, yet amplified responses – a hallmark of neural responses throughout the brain [194, 195, 196]. In a model of primary visual cortex [197, 198], EPI reveals how the recurrent processing across different neuron-type populations shapes excitatory variability: a finding that we show is analytically intractable. Finally, we investigated the possible connectivities of a superior colliculus model that allow execution of different tasks on interleaved trials [199]. EPI discovered a rich distribution containing two connectivity regimes with different solution classes. We queried the deep probability distribution learned by EPI to produce a mechanistic understanding of neural responses in each regime. Intriguingly, the inferred connectivities of each regime reproduced results from optogenetic inactivation experiments in markedly different ways. These theoretical insights afforded by EPI illustrate the value of deep inference for the interrogation of neural circuit models.

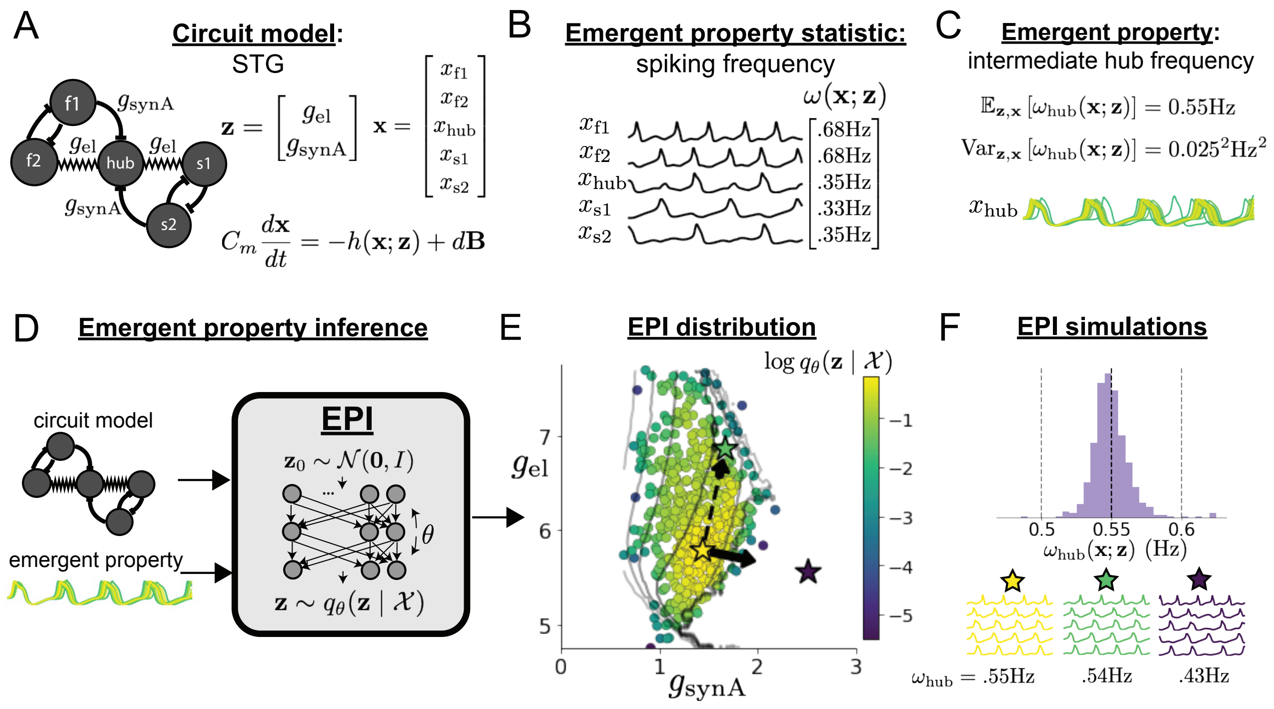
4.2 Results

4.2.1 Motivating emergent property inference of theoretical models

Consideration of the typical workflow of theoretical modeling clarifies the need for emergent property inference. First, one designs or chooses an existing circuit model that, it is hypothesized, captures the computation of interest. To ground this process in a well-known example, consider the stomatogastric ganglion (STG) of crustaceans, a small neural circuit which generates multiple rhythmic muscle activation patterns for digestion [45]. Despite full knowledge of STG connectivity and a precise characterization of its rhythmic pattern generation, biophysical models of the STG have complicated relationships between circuit parameters and computation [193, 173].

A subcircuit model of the STG [41] is shown schematically in Figure 4.1A. The fast population (f1 and f2) represents the subnetwork generating the pyloric rhythm and the slow population (s1 and s2) represents the subnetwork of the gastric mill rhythm. The two fast neurons mutually inhibit one another, and spike at a greater frequency than the mutually inhibiting slow neurons. The hub neuron couples with either the fast or slow population, or both depending on modulatory conditions. The jagged connections indicate electrical coupling having electrical conductance g_{el}, smooth connections in the diagram are inhibitory synaptic projections having strength g_{synA} onto the hub neuron, and $g_{\text{synB}} = 5\text{nS}$ for mutual inhibitory connections. Note that the behavior of this model will be critically dependent on its parameterization – the choices of conductance parameters $\mathbf{z} = [g_{\text{el}}, g_{\text{synA}}]$.

Second, once the model is selected, one must specify what the model should produce. In this STG model, we are concerned with neural spiking frequency, which emerges from the dynamics of the circuit model (Fig. 4.1B). An emergent property studied by Gutierrez et al. is the hub neuron firing at an intermediate frequency between the intrinsic spiking rates of the fast and slow



A
Circuit model:
STG
f1
f2
hub
s1
s2
g_{synA}
g_{el}
$\mathbf{z} = \begin{bmatrix} g_{\text{el}} \\ g_{\text{synA}} \end{bmatrix}$
$\mathbf{x} = \begin{bmatrix} x_{\text{f1}} \\ x_{\text{f2}} \\ x_{\text{hub}} \\ x_{\text{s1}} \\ x_{\text{s2}} \end{bmatrix}$
$C_m \frac{d\mathbf{x}}{dt} = -h(\mathbf{x};\mathbf{z}) + d\mathbf{B}$
B
Emergent property statistic:
spiking frequency
$\omega(\mathbf{x};\mathbf{z})$
x_{f1} .68Hz
x_{f2} .68Hz
x_{hub} .35Hz
x_{s1} .33Hz
x_{s2} .35Hz
C
Emergent property:
intermediate hub frequency
$\mathbb{E}_{\mathbf{z},\mathbf{x}}[\omega_{\text{hub}}(\mathbf{x};\mathbf{z})] = 0.55\text{Hz}$
$\text{Var}_{\mathbf{z},\mathbf{x}}[\omega_{\text{hub}}(\mathbf{x};\mathbf{z})] = 0.025^2\text{Hz}^2$
x_{hub}
D
Emergent property inference
circuit model
emergent property
EPI
$\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, I)$
θ
$\mathbf{z} \sim q_\theta(\mathbf{z} \mid \mathcal{X})$
E
EPI distribution
$\log q_\theta(\mathbf{z} \mid \mathcal{X})$
g_{el}
g_{synA}
F
EPI simulations
$\omega_{\text{hub}}(\mathbf{x};\mathbf{z})$ (Hz)
$\omega_{\text{hub}} = .55\text{Hz}$
.54Hz
.43Hz

Figure 4.1: Emergent property inference in the stomatogastric ganglion. **A**. Conductance-based subcircuit model of the STG. **B**. Spiking frequency $\omega(\mathbf{x};\mathbf{z})$ is an emergent property statistic. Simulated at $g_{\text{el}} = 4.5\text{nS}$ and $g_{\text{synA}} = 3\text{nS}$. **C**. The emergent property of intermediate hub frequency. Simulated activity traces are colored by log probability of generating parameters in the EPI distribution (Panel E). **D**. For a choice of circuit model and emergent property, EPI learns a deep probability distribution of parameters $\mathbf{z}$. **E**. The EPI distribution producing intermediate hub frequency. Samples are colored by log probability density. Contours of hub neuron frequency error are shown at levels of .525, .53, ... .575 Hz (dark to light gray away from mean). Dimension of sensitivity $\mathbf{v}_1$ (solid arrow) and robustness $\mathbf{v}_2$ (dashed arrow). **F** (Top) The predictions of the EPI distribution. The black and gray dashed lines show the mean and two standard deviations according the emergent property. (Bottom) Simulations at the starred parameter values.

populations. This emergent property (EP) is shown in Figure 4.1C at an average frequency of 0.55Hz. To be precise, we define intermediate hub frequency not strictly as 0.55Hz, but frequencies of moderate deviation from 0.55Hz between the fast (.35Hz) and slow (.68Hz) frequencies.

Third, the model parameters producing the emergent property are inferred. By precisely quantifying the emergent property of interest as a statistical feature of the model, we use emergent property inference (EPI) to condition directly on this emergent property. Before presenting technical details (in the following section), let us understand emergent property inference schematically. EPI (Fig. 4.1D) takes, as input, the model and the specified emergent property, and as its output, returns the parameter distribution (Fig. 4.1E). This distribution – represented for clarity as samples from the distribution – is a parameter distribution constrained such that the circuit model produces the emergent property. Once EPI is run, the returned distribution can be used to efficiently generate additional parameter samples. Most importantly, the inferred distribution can be efficiently queried to quantify the parametric structure that it captures. By quantifying the parametric structure governing the emergent property, EPI informs the central question of this inverse problem: what aspects or combinations of model parameters have the desired emergent property?

4.2.2 Emergent property inference via deep generative models

EPI formalizes the three-step procedure of the previous section with deep probability distributions [68, 69]. First, as is typical, we consider the model as a coupled set of noisy differential equations. In this STG example, the model activity (or state) $\mathbf{x} = [x_{\text{f1}}, x_{\text{f2}}, x_{\text{hub}}, x_{\text{s1}}, x_{\text{s2}}]$ is the membrane potential for each neuron, which evolves according to the biophysical conductance-based equation:

$$C_m \frac{d\mathbf{x}(t)}{dt} = -h(\mathbf{x}(t); \mathbf{z}) + d\mathbf{B} \tag{4.1}$$

where C_m=1nF, and $\mathbf{h}$ is a sum of the leak, calcium, potassium, hyperpolarization, electrical, and synaptic currents, all of which have their own complicated dependence on activity $\mathbf{x}$ and parameters

$\mathbf{z} = [g_{\text{el}}, g_{\text{synA}}]$, and $d\mathbf{B}$ is white gaussian noise [41] (see Section 4.4.2.1 for more detail).

Second, we determine that our model should produce the emergent property of "intermediate hub frequency" (Figure 4.1C). We stipulate that the hub neuron's spiking frequency – denoted by statistic $\omega_{\text{hub}}(\mathbf{x})$ – is close to a frequency of 0.55Hz, between that of the slow and fast frequencies. Mathematically, we define this emergent property with two constraints: that the mean hub frequency is 0.55Hz,

$$\mathbb{E}_{\mathbf{z},\mathbf{x}}\left[\omega_{\text{hub}}(\mathbf{x};\mathbf{z})\right] \;=\; 0.55 \tag{4.2}$$

and that the variance of the hub frequency is moderate

$$\text{Var}_{\mathbf{z},\mathbf{x}}\left[\omega_{\text{hub}}(\mathbf{x};\mathbf{z})\right] \;=\; 0.025^2. \tag{4.3}$$

In the emergent property of intermediate hub frequency, the statistic of hub neuron frequency is an expectation over the distribution of parameters $\mathbf{z}$ and the distribution of the data $\mathbf{x}$ that those parameters produce. We define the emergent property $\mathcal{X}$ as the collection of these two constraints. In general, an emergent property is a collection of constraints on statistical moments that together define the computation of interest.

Third, we perform emergent property inference: we find a distribution over parameter configurations $\mathbf{z}$ of models that produce the emergent property; in other words, they satisfy the constraints introduced in Equations 4.2 and 4.3. This distribution will be chosen from a family of probability distributions $\mathcal{Q} = \{q_{\boldsymbol{\theta}}(\mathbf{z}) : \boldsymbol{\theta} \in \Theta\}$, defined by a deep neural network [68, 69] (Figure 4.1D, EPI box). Deep probability distributions map a simple random variable $\mathbf{z}_0$ (e.g. an isotropic gaussian) through a deep neural network with weights and biases $\boldsymbol{\theta}$ to parameters $\mathbf{z} = g_{\boldsymbol{\theta}}(\mathbf{z}_0)$ of a suitably complicated distribution (see Section 4.4.1.2 for more details). Many distributions in $\mathcal{Q}$ will respect the emergent property constraints, so we select the most random (highest entropy) distribution, which also means this approach is equivalent to bayesian variational inference (see Section 4.4.1.6). In EPI optimization, stochastic gradient steps in $\boldsymbol{\theta}$ are taken such that entropy is maximized, and the emergent property $\mathcal{X}$ is produced (see Section 4.4.1). We then denote the inferred EPI distribution as $q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathcal{X})$, since the structure of the learned parameter distribution is determined by weights and biases $\boldsymbol{\theta}$, and this distribution is conditioned upon emergent property $\mathcal{X}$.

The structure of the inferred parameter distributions of EPI can be analyzed to reveal key information about how the circuit model produces the emergent property. As probability in the EPI distribution decreases away from the mode of $q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathcal{X})$ (Fig. 4.1E yellow star), the emergent property deteriorates. Perturbing $\mathbf{z}$ along a dimension in which $q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathcal{X})$ changes little will not disturb the emergent property, making this parameter combination *robust* with respect to the emergent property. In contrast, if $\mathbf{z}$ is perturbed along a dimension with strongly decreasing $q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathcal{X})$, that parameter combination is deemed *sensitive* [182, 185]. By querying the second order derivative (Hessian) of $\log q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathcal{X})$ at a mode, we can quantitatively identify how sensitive (or robust) each eigenvector is by its eigenvalue; the more negative, the more sensitive and the closer to zero, the more robust (see Section 4.4.2.4). Indeed, samples equidistant from the mode along these dimensions of sensitivity ($\mathbf{v}_1$, smaller eigenvalue) and robustness ($\mathbf{v}_2$, greater eigenvalue) (Fig. 4.1E, arrows) agree with error contours (Fig. 4.1E contours) and have diminished or preserved hub frequency, respectively (Fig. 4.1F activity traces). The directionality of $\mathbf{v}_2$ suggests that changes in conductance along this parameter combination will most preserve hub neuron firing between the intrinsic rates of the pyloric and gastric mill rhythms. Importantly and unlike alternative techniques, once an EPI distribution has been learned, the modes and Hessians of the distribution can be measured with trivial computation (see Section 4.4.1.2).

In the following sections, we demonstrate EPI on three neural circuit models across ranges of biological realism, neural system function, and network scale. First, we demonstrate the superior scalability of EPI compared to alternative techniques by inferring high-dimensional distributions of recurrent neural network connectivities that exhibit amplified, yet stable responses. Next, in a model of primary visual cortex [197, 198], we show how EPI discovers parametric degeneracy, revealing how input variability across neuron types affects the excitatory population. Finally, in a model of superior colliculus [199], we used EPI to capture multiple parametric regimes of task switching, and queried the dimensions of parameter sensitivity to characterize each regime.

4.2.3 Scaling inference of recurrent neural network connectivity with EPI

To understand how EPI scales in comparison to existing techniques, we consider recurrent neural networks (RNNs). Transient amplification is a hallmark of neural activity throughout cortex, and is often thought to be intrinsically generated by recurrent connectivity in the responding cortical area [194, 195, 196]. It has been shown that to generate such amplified, yet stabilized responses, the connectivity of RNNs must be non-normal [200, 194], and satisfy additional constraints [201]. In theoretical neuroscience, RNNs are optimized and then examined to show how dynamical systems could execute a given computation [43, 46], but such biologically realistic constraints on connectivity [200, 194, 201] are ignored for simplicity or because constrained optimization is difficult. In general, access to distributions of connectivity that produce theoretical criteria like stable amplification, chaotic fluctuations [168], or low tangling [73] would add scientific value to existing research with RNNs. Here, we use EPI to learn RNN connectivities producing stable amplification, and demonstrate the superior scalability and efficiency of EPI to alternative approaches.

We consider a rank-2 RNN with N neurons having connectivity $W = UV^\top$ and dynamics

$$\tau \dot{\mathbf{x}} = -\mathbf{x} + W\mathbf{x}, \tag{4.4}$$

where $U = \begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_2 \end{bmatrix} + g\chi^{(U)}$, $V = \begin{bmatrix} \mathbf{V}_1 & \mathbf{V}_2 \end{bmatrix} + g\chi^{(V)}$, $\mathbf{U}_1\mathbf{U}_2, \mathbf{V}_1, \mathbf{V}_2 \in [-1, 1]^N$, and $\chi^{(U)}_{i,j}, \chi^{(V)}_{i,j} \sim \mathcal{N}(0, 1)$. We infer connectivity parameters $\mathbf{z} = [\mathbf{U}_1, \mathbf{U}_2, \mathbf{V}_1, \mathbf{V}_2]$ that produce stable amplification. Two conditions are necessary and sufficient for RNNs to exhibit stable amplification [201]: $\text{real}(\lambda_1) < 1$ and $\lambda^s_1 > 1$, where λ_1 is the eigenvalue of W with greatest real part and λ^s is the maximum eigenvalue of $W^s = \frac{W+W^\top}{2}$. RNNs with $\text{real}(\lambda_1) = 0.5 \pm 0.5$ and $\lambda^s_1 = 1.5 \pm 0.5$ will be stable with modest decay rate ($\text{real}(\lambda_1)$ close to its upper bound of 1) and exhibit modest amplification

(λ_1^s close to its lower bound of 1). EPI can naturally condition on this emergent property

$$\begin{aligned} \mathcal{X} \ : \ \mathbb{E}_{\mathbf{z},\mathbf{x}} \begin{bmatrix} \text{real}(\lambda_1) \\ \lambda_1^s \end{bmatrix} &= \begin{bmatrix} 0.5 \\ 1.5 \end{bmatrix} \\ \text{Var}_{\mathbf{z},\mathbf{x}} \begin{bmatrix} \text{real}(\lambda_1) \\ \lambda_1^s \end{bmatrix} &= \begin{bmatrix} 0.25^2 \\ 0.25^2 \end{bmatrix}. \end{aligned} \tag{4.5}$$

Variance constraints predicate that the majority of the distribution (within two standard deviations) are within the specified ranges.

For comparison, we infer the parameters $\mathbf{z}$ likely to produce stable amplification using two alternative simulation-based inference approaches. Sequential Monte Carlo approximate bayesian computation (SMC-ABC) [52] is a rejection sampling approach that uses SMC techniques to improve efficiency, and sequential neural posterior estimation (SNPE) [190] approximates posteriors with deep probability distributions (see Section 4.4.1.1). Unlike EPI, these statistical inference techniques do not constrain the predictions of the inferred distribution, so they were run by conditioning on an exemplar dataset $\mathbf{x}_0 = \boldsymbol{\mu}$, following standard practice with these methods [52, 190]. To compare the efficiency of these different techniques, we measured the time and number of simulations necessary for the distance of the predictive mean to be less than 0.5 from $\boldsymbol{\mu} = \mathbf{x}_0$ (see Section 4.4.3).

As the number of neurons N in the RNN, and thus the dimension of the parameter space $\mathbf{z} \in [-1, 1]^{4N}$, is scaled, we see that EPI converges at greater speed and at greater dimension than SMC-ABC and SNPE (Fig. 4.2A). It also becomes most efficient to use EPI in terms of simulation count at $N = 50$ (Fig. 4.2B). It is well known that ABC techniques struggle in parameter spaces of modest dimension [202], yet we were careful to assess the scalability of SNPE, which is a more closely related methodology to EPI. Between EPI and SNPE, we closely controlled the number of parameters in deep probability distributions by dimensionality (Fig. 4.7), and tested more aggressive SNPE hyperparameter choices when SNPE failed to converge (Fig. 4.8). In this analysis, we see that deep inference techniques EPI and SNPE are far more amenable to inference

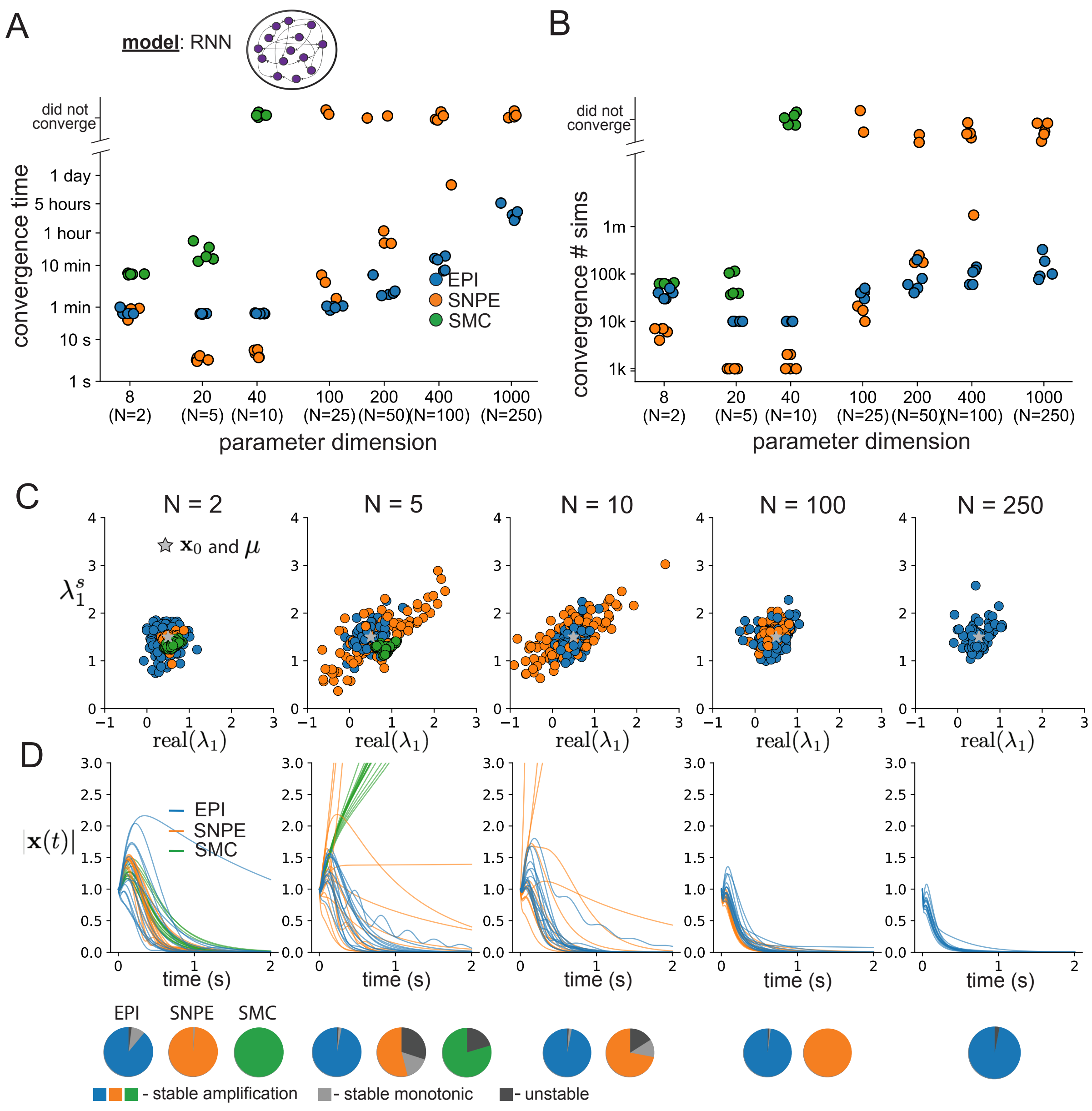


Figure 4.2: **A**. Wall time of EPI (blue), SNPE (orange), and SMC-ABC (green) to converge on RNN connectivities producing stable amplification. Each dot shows convergence time for an individual random seed. For reference, the mean wall time for EPI to achieve its full constraint convergence (means and variances) is shown (blue line). **B**. Simulation count of each algorithm to achieve convergence. Same conventions as A. **C**. The predictive distributions of connectivities inferred by EPI (blue), SNPE (orange), and SMC-ABC (green), with reference to $\mathbf{x}_0 = \boldsymbol{\mu}$ (gray star). **D**. Simulations of networks inferred by each method ($\tau = 100ms$). Each trace (15 per algorithm) corresponds to simulation of one z. (Below) Ratio of obtained samples producing stable amplification, stable monotonic decay, and instability.

of high dimensional RNN connectivities than rejection sampling techniques like SMC-ABC, and that EPI outperforms SNPE in both wall time (elapsed real time) and simulation count.

No matter the number of neurons, EPI always produces connectivity distributions with mean and variance of real(λ_1) and λ_1^s according to $\mathcal{X}$ (Fig. 4.2C, blue). For the dimensionalities in which SMC-ABC is tractable, the inferred parameters are concentrated and offset from the exemplar dataset $\mathbf{x}_0$ (Fig. 4.2C, green). When using SNPE, the predictions of the inferred parameters are highly concentrated at some RNN sizes and widely varied in others (Fig. 4.2C, orange). We see these properties reflected in simulations from the inferred distributions: EPI produces a consistent variety of stable, amplified activity norms $|\mathbf{x}(t)|$, SMC-ABC produces a limited variety of responses, and the changing variety of responses from SNPE emphasizes the control of EPI on parameter predictions (Fig. 4.2D). Even for moderate neuron counts, the predictions of the inferred distribution of SNPE are highly dependent on N and g, while EPI maintains the emergent property across choices of RNN (see Section 4.4.3.5).

To understand these differences, note that EPI outperforms SNPE in high dimensions by using gradient information (from $\nabla_{\mathbf{z}}[\text{real}(\lambda_1), \lambda_1^s]^\top$). This choice agrees with recent speculation that such gradient information could improve the efficiency of simulation-based inference techniques [50], as well as reflecting the classic tradeoff between gradient-based and sampling-based estimators (scaling and speed versus generality). Since gradients of the emergent property are necessary in EPI optimization, gradient tractability is a key criteria when determining the suitability of a simulation-based inference technique. If the emergent property gradient is efficiently calculated, EPI is a clear choice for inferring high dimensional parameter distributions.

4.3 Discussion

In neuroscience, machine learning has primarily been used to reveal structure in neural datasets [17]. Careful inference procedures are developed for these statistical models allowing precise, quantitative reasoning, which clarifies the way data informs beliefs about the model parameters. However, these statistical models often lack resemblance to the underlying biology, making it unclear how to go from the structure revealed by these methods, to the neural mechanisms giving rise to it. In contrast, theoretical neuroscience has primarily focused on careful models of neural circuits and the production of emergent properties of computation, rather than measuring structure in neural datasets. In this work, we improve upon parameter inference techniques in theoretical neuroscience with emergent property inference, harnessing deep learning towards parameter inference in neural circuit models (see Section 4.4.1.1).

Methodology for statistical inference in circuit models has evolved considerably in recent years. Early work used rejection sampling techniques [51, 181, 52], but EPI and another recently developed methodology [190] employ deep learning to improve efficiency and provide flexible approximations. SNPE has been used for posterior inference of parameters in circuit models conditioned upon exemplar data used to represent computation, but it does not infer parameter distributions that only produce the computation of interest like EPI (see Section 4.2.3). When strict control over the predictions of the inferred parameters is necessary, EPI uses a constrained optimization technique [191] (see Section 4.4.1.4) to make inference conditioned on the emergent property possible.

A key difference between EPI and SNPE, is that EPI uses gradients of the emergent property throughout optimization. In Section 4.2.3, we showed that such gradients confer beneficial scaling properties, but a concern remains that emergent property gradients may be too computationally intensive. Even in a case of close biophysical realism with an expensive emergent property gradient, EPI was run successfully on intermediate hub frequency in a 5-neuron subcircuit model of the STG (Section 4.2.1). However, conditioning on the pyloric rhythm [203] in a model of the pyloric subnetwork model [173] proved to be prohibitive with EPI. The pyloric subnetwork requires many time steps for simulation and many key emergent property statistics (e.g. burst duration and phase gap) are not calculable or easily approximated with differentiable functions. In such cases, SNPE, which does not require differentiability of the emergent property, has proven useful [190]. In summary, choice of deep inference technique should consider emergent property complexity and differentiability, dimensionality of parameter space, and the importance of constraining the model behavior predicted by the inferred parameter distribution.

Acknowledgements:

This work was funded by NSF Graduate Research Fellowship, DGE-1644869, McKnight Endowment Fund, NIH NINDS 5R01NS100066, Simons Foundation 542963, NSF NeuroNex Award, DBI-1707398, The Gatsby Charitable Foundation, Simons Collaboration on the Global Brain Postdoctoral Fellowship, Chinese Postdoctoral Science Foundation, and International Exchange Program Fellowship. We also acknowledge the Marine Biological Laboratory Methods in Computational Neuroscience Course, where this work was discussed and explored in its early stages. Helpful conversations were had with Larry Abbott, Stephen Baccus, James Fitzgerald, Gabrielle Gutierrez, Francesca Mastrogiuseppe, Srdjan Ostojic, Liam Paninski, and Dhruva Raman.

Data availability statement:

The datasets generated during this study have been made publicly available on Zenodo at this address: https://doi.org/10.5281/zenodo.4910010 .

Code availability statement:

All software written for the current study is available at https://github.com/cunningham-lab/epi.

4.4 Methods

4.4.1 Emergent property inference (EPI)

Solving inverse problems is an important part of theoretical neuroscience, since we must understand how neural circuit models and their parameter choices produce computations. Recently, research on machine learning methodology for neuroscience has focused on finding latent structure in large-scale neural datasets, while research in theoretical neuroscience generally focuses on developing precise neural circuit models that can produce computations of interest. By quantifying computation into an *emergent property* through statistics of the emergent activity of neural circuit models, we can adapt the modern technique of deep probabilistic inference towards solving inverse problems in theoretical neuroscience. Here, we introduce a novel method for statistical inference, which uses deep networks to learn parameter distributions constrained to produce emergent properties of computation.

Consider model parameterization $\mathbf{z}$, which is a collection of scientifically meaningful variables that govern the complex simulation of data $\mathbf{x}$. For example (see Section 4.2.1), $\mathbf{z}$ may be the electrical conductance parameters of an STG subcircuit, and $\mathbf{x}$ the evolving membrane potentials of the five neurons. In terms of statistical modeling, this circuit model has an intractable likelihood $p(\mathbf{x} \mid \mathbf{z})$, which is predicated by the stochastic differential equations that define the model. From a theoretical perspective, we are less concerned about the likelihood of an exemplar dataset $\mathbf{x}$, but rather the emergent property of intermediate hub frequency (which implies a consistent dataset $\mathbf{x}$).

In this work, emergent properties $\mathcal{X}$ are defined through the choice of emergent property statistic $f(\mathbf{x}; \mathbf{z})$ (which is a vector of one or more statistics), and its means $\boldsymbol{\mu}$, and variances $\boldsymbol{\sigma}^2$:

$$\mathcal{X} : \ \mathbb{E}_{\mathbf{z},\mathbf{x}}\left[f(\mathbf{x}; \mathbf{z})\right] = \boldsymbol{\mu}, \ \ \mathrm{Var}_{\mathbf{z},\mathbf{x}}\left[f(\mathbf{x}; \mathbf{z})\right] = \boldsymbol{\sigma}^2. \tag{4.6}$$

In general, an emergent property may be a collection of first-, second-, or higher-order moments of a group of statistics, but this study focuses on the case written in Equation 4.6. In the STG example, intermediate hub frequency is defined by mean and variance constraints on the statistic of hub neuron frequency $\omega_{\text{hub}}(\mathbf{x}; \mathbf{z})$ (Equations 4.2 and 4.3). Precisely, the emergent property statistics $f(\mathbf{x}; \mathbf{z})$ must have means $\boldsymbol{\mu}$ and variances $\boldsymbol{\sigma}^2$ over the EPI distribution of parameters ($\mathbf{z} \sim q_{\boldsymbol{\theta}}(\mathbf{z})$) and the data produced by those parameters ($\mathbf{x} \sim p(\mathbf{x} \mid \mathbf{z})$), where the inferred parameter distribution $q_{\boldsymbol{\theta}}(\mathbf{z})$ itself is parameterized by deep network weights and biases $\boldsymbol{\theta}$.

In EPI, a deep probability distribution $q_{\boldsymbol{\theta}}(\mathbf{z})$ is optimized to approximate the parameter distribution producing the emergent property $\mathcal{X}$. In contrast to simpler classes of distributions like the gaussian or mixture of gaussians, deep probability distributions are far more flexible and capable of fitting rich structure [68, 69]. In deep probability distributions, a simple random variable $\mathbf{z}_0 \sim q_0(\mathbf{z}_0)$ (we choose an isotropic gaussian) is mapped deterministically via a sequence of deep neural network layers (g_1, .. g_l) parameterized by weights and biases $\boldsymbol{\theta}$ to the support of the distribution of interest:

$$\mathbf{z} = g_{\boldsymbol{\theta}}(\mathbf{z}_0) = g_l(..g_1(\mathbf{z}_0)) \sim q_{\boldsymbol{\theta}}(\mathbf{z}). \tag{4.7}$$

Such deep probability distributions embed the inferred distribution in a deep network. Once optimized, this deep network representation of a distribution has remarkably useful properties: fast sampling and probability evaluations. Importantly, fast probability evaluations confer fast gradient and Hessian calculations as well.

Given this choice of circuit model and emergent property $\mathcal{X}$, $q_{\boldsymbol{\theta}}(\mathbf{z})$ is optimized via the neural network parameters $\boldsymbol{\theta}$ to find a maximally entropic distribution $q_{\boldsymbol{\theta}}^*$ within the deep variational family $Q = \{q_{\boldsymbol{\theta}}(\mathbf{z}) : \boldsymbol{\theta} \in \Theta\}$ that produces the emergent property $\mathcal{X}$:

$$\begin{aligned} q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathcal{X}) = q_{\boldsymbol{\theta}}^*(\mathbf{z}) &= \underset{q_{\boldsymbol{\theta}} \in Q}{\operatorname{argmax}}\, H(q_{\boldsymbol{\theta}}(\mathbf{z})) \\ \text{s.t. } \mathcal{X} \ &: \ \mathbb{E}_{\mathbf{z},\mathbf{x}}\left[f(\mathbf{x}; \mathbf{z})\right] = \boldsymbol{\mu}, \mathrm{Var}_{\mathbf{z},\mathbf{x}}\left[f(\mathbf{x}; \mathbf{z})\right] = \boldsymbol{\sigma}^2, \end{aligned} \tag{4.8}$$

where $H(q_{\boldsymbol{\theta}}(\mathbf{z})) = \mathbb{E}_{\mathbf{z}}\left[-\log q_{\boldsymbol{\theta}}(z)\right]$ is entropy. By maximizing the entropy of the inferred distribution $q_{\boldsymbol{\theta}}$, we select the most random distribution in family Q that satisfies the constraints of the emergent property. Since entropy is maximized in Equation 4.8, EPI is equivalent to bayesian variational inference (see Section 4.4.1.6), which is why we specify the inferred distribution of EPI as conditioned upon emergent property $\mathcal{X}$ with the notation $q_\theta(\mathbf{z} \mid \mathcal{X})$. To run this constrained optimization, we use an augmented lagrangian objective, which is the standard approach for constrained optimization [204], and the approach taken to fit Maximum Entropy Flow Networks (MEFNs) [191]. This procedure is detailed in Section 4.4.1.4 and the pseudocode in Algorithm 1.

In the remainder of Section 4.4.1, we will explain the finer details and motivation of the EPI method. First, we explain related approaches and what EPI introduces to this domain (Section 4.4.1.1). Second, we describe the special class of deep probability distributions used in EPI called normalizing flows (Section 4.4.1.2). Then, we establish the known relationship between maximum entropy distributions and exponential families (Section 4.4.1.3). Next, we explain the constrained optimization technique used to solve Equation 4.8 (Section 4.4.1.4). Then, we demonstrate the details of this optimization in a toy example (Section 4.4.1.5). Finally, we explain how EPI is equivalent to variational inference (Section 4.4.1.6).

4.4.1.1 Related approaches

When bayesian inference problems lack conjugacy, scientists use approximate inference methods like variational inference (VI) [205] and Markov chain Monte Carlo (MCMC) [206, 207]. After optimization, variational methods return a parameterized posterior distribution, which we can analyze. Also, the variational approximation is often chosen such that it permits fast sampling. In contrast MCMC methods only produce samples from the approximated posterior distribution. No parameterized distribution is estimated, and additional samples are always generated with the same sampling complexity. Inference in models defined by systems of differential equations has been demonstrated with MCMC [208], although this approach requires tractable likelihoods. Advancements have introduced sampling [209], likelihood approximation [210], and uncertainty quantification techniques [211] to make MCMC approaches more efficient and expand the class of applicable models.

Simulation-based inference [50] is model parameter inference in the absence of a tractable likelihood function. The most prevalent approach to simulation-based inference is approximate bayesian computation (ABC) [51], in which satisfactory parameter samples are kept from random prior sampling according to a rejection heuristic. The obtained set of parameters do not have a probabilities, and further insight about the model must be gained from examination of the parameter set and their generated activity. Methodological advances to ABC methods have come through the use of Markov chain Monte Carlo (MCMC-ABC) [181] and sequential Monte Carlo (SMC-ABC) [52] sampling techniques. SMC-ABC is considered state-of-the-art ABC, yet this approach still struggles to scale in dimensionality [202] (cf. Fig. 4.2). Still, this method has enjoyed much success in systems biology [212]. Furthermore, once a parameter set has been obtained by SMC-ABC from a finite set of particles, the SMC-ABC algorithm must be run again from scratch with a new population of initialized particles to obtain additional samples.

For scientific model analysis, we seek a parameter distribution represented by an approximating distribution as in variational inference [205]: a variational approximation that once optimized yields fast analytic calculations and samples. For the reasons described above, ABC and MCMC techniques are not suitable, since they only produce a set of parameter samples lacking probabilities and have unchanging sampling rate. EPI infers parameters in circuit models using the MEFN [191] algorithm with a deep variational approximation. The deep neural network of EPI (Fig. 4.1E) defines the parametric form (with weights and biases as variational parameters $\boldsymbol{\theta}$) of the variational approximation of the inferred parameter distribution $q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})$. The EPI optimization is enabled using stochastic gradient techniques in the spirit of likelihood-free variational inference [189]. The analytic relationship between EPI and variational inference is explained in Section 4.4.1.6.

We note that, during our preparation and early presentation of this work [213, 214], another work has arisen with broadly similar goals: bringing statistical inference to mechanistic models of neural circuits [215, 216, 190]. We are encouraged by this general problem being recognized by others in the community, and we emphasize that these works offer complementary neuroscientific contributions (different theoretical models of focus) and use different technical methodologies (ours is built on our prior work [191], theirs similarly [217]).

The method EPI differs from SNPE in some key ways. SNPE belongs to a “sequential" class of recently developed simulation-based inference methods in which two neural networks are used for posterior inference. This first neural network is a deep probability distribution (normalizing flow) used to estimate the posterior $p(\mathbf{z} \mid \mathbf{x})$ (SNPE) or the likelihood $p(\mathbf{x} \mid \mathbf{z})$ (sequential neural likelihood (SNL) [218]). A recent approach uses an unconstrained neural network to estimate the likelihood ratio (sequential neural ratio estimation (SNRE) [219]). In SNL and SNRE, MCMC sampling techniques are used to obtain samples from the approximated posterior. This contrasts with EPI and SNPE, which use deep probability distributions to model parameters, which facilitates immediate measurements of sample probability, gradient, or Hessian for system analysis. The second neural network in this sequential class of methods is the amortizer. This unconstrained deep network maps data $\mathbf{x}$ (or statistics $f(\mathbf{x}; \mathbf{z})$ or model parameters $\mathbf{z}$) to the weights and biases of the first neural network. These methods are optimized on a conditional density (or ratio) estimation objective. The data used to optimize this objective are generated via an adaptive procedure, in which training data pairs $(\mathbf{x}_i, \mathbf{z}_i)$ become sequentially closer to the true data and posterior.

The approximating fidelity of the deep probability distribution in sequential approaches is optimized to generalize across the training distribution of the conditioning variable. This generalization property of the sequential methods can reduce the accuracy at the singular posterior of interest. Whereas in EPI, the entire expressivity of the deep probability distribution is dedicated to learning a single distribution as well as possible. The well-known inverse mapping problem of exponential families [137] prohibits an amortization-based approach in EPI, since EPI learns an exponential family distribution parameterized by its mean (in contrast to its natural parameter, see Section 4.4.1.3). However, we have shown that the same two-network architecture of the sequential simulation-based inference methods can be used for amortized inference in intractable exponential family posteriors when using their natural parameterization [124].

Finally, one important differentiating factor between EPI and sequential simulation-based inference methods is that EPI leverages gradients $\nabla_{\mathbf{z}} f(\mathbf{x}; \mathbf{z})$ during optimization. These gradients can improve convergence time and scalability, as we have shown on an example conditioning low-rank RNN connectivity on the property of stable amplification (see Section 4.2.3). With EPI, we prove out the suggestion that a deep inference technique can improve efficiency by leveraging these emergent property gradients when they are tractable. Sequential simulation-based inference techniques may be better suited for scientific problems where $\nabla_{\mathbf{z}} f(\mathbf{x};\mathbf{z})$ is intractable or unavailable, like when there is a nondifferentiable emergent property. However, the sequential simulation-based inference techniques cannot constrain the predictions of the inferred distribution in the manner of EPI.

Structural identifiability analysis involves the measurement of sensitivity and unidentifiabilities in scientific models. Around a single parameter choice, one can measure the Jacobian. One approach for this calculation that scales well is EAR [183]. A popular efficient approach for systems of ODEs has been neural ODE adjoint [220] and its stochastic adaptation [221]. Casting identifiability as a statistical estimation problem, the profile likelihood works via iterated optimization while holding parameters fixed [182]. An exciting recent method is capable of recovering the functional form of such unidentifiabilities away from a point by following degenerate dimensions of the fisher information matrix [185]. Global structural non-identifiabilities can be found for models with polynomial or rational dynamics equations using DAISY [222], or through mean optimal transformations [223]. With EPI, we have all the benefits given by a statistical inference method plus the ability to query the first- or second-order gradient of the probability of the inferred distribution at any chosen parameter value. The second-order gradient of the log probability (the Hessian), which is directly afforded by EPI distributions, produces quantified information about parametric sensitivity of the emergent property in parameter space (see Section 4.2.2).

4.4.1.2 Deep probability distributions and normalizing flows

Deep probability distributions are comprised of multiple layers of fully connected neural networks (Equation 4.7). When each neural network layer is restricted to be a bijective function, the sample density can be calculated using the change of variables formula at each layer of the network. For $\mathbf{z}_i = g_i(\mathbf{z_{i-1}})$,

$$p(\mathbf{z}_i) = p(g_i^{-1}(\mathbf{z}_i)) \left| \det \frac{\partial g_i^{-1}(\mathbf{z}_i)}{\partial \mathbf{z}_i} \right| = p(\mathbf{z}_{i-1}) \left| \det \frac{\partial g_i(\mathbf{z}_{i-1})}{\partial \mathbf{z}_{i-1}} \right|^{-1}. \tag{4.9}$$

However, this computation has cubic complexity in dimensionality for fully connected layers. By restricting our layers to normalizing flows [68, 69] – bijective functions with fast log determinant Jacobian computations, we obtain a fast, tractable calculation of the sample log probability. Fast log probability calculation confers efficient optimization of the maximum entropy objective (see Section 4.4.1.4).

We use the real NVP [72] normalizing flow class, because its coupling architecture confers both fast sampling (forward) and fast log probability evaluation (backward). Fast probability evaluation facilitates fast gradient and Hessian evaluation of log probability throughout parameter space. Glow permutations were used in between coupling stages [192]. This is in contrast to autoregressive architectures [71, 70], in which only one of the forward or backward passes can be efficient. In this work, normalizing flows are used as flexible parameter distribution approximations $q_{\boldsymbol{\theta}}(\mathbf{z})$ having weights and biases $\boldsymbol{\theta}$. We specify the architecture used in each application by the number of real NVP affine coupling stages, and the number of neural network layers and units per layer of the conditioning functions.

When calculating Hessians of log probabilities in deep probability distributions, it is important to consider the normalizing flow architecture. With autoregressive architectures [70, 71], fast sampling and fast log probability evaluations are mutually exclusive. That makes these architectures undesirable for EPI, where efficient sampling is important for optimization, and log probability evaluation speed predicates the efficiency of gradient and Hessian calculations. With real NVP coupling architectures, we get both fast sampling and fast Hessians making both optimization and scientific analysis efficient.

4.4.1.3 Maximum entropy distributions and exponential families

The inferred distribution of EPI is a maximum entropy distribution, which have fundamental links to exponential family distributions. A maximum entropy distribution of form:

$$\begin{aligned} p^*(\mathbf{z}) &= \underset{p \in \mathcal{P}}{\operatorname{argmax}}\, H(p(\mathbf{z})) \\ &\text{s.t. } \mathbb{E}_{\mathbf{z}\sim p}\left[T(\mathbf{z})\right] = \boldsymbol{\mu}_{\text{opt}}, \end{aligned} \tag{4.10}$$

where $T(\mathbf{z})$ is the sufficient statistics vector and $\boldsymbol{\mu}_{\text{opt}}$ a vector of their mean values, will have probability density in the exponential family:

$$p^*(\mathbf{z}) \propto \exp(\boldsymbol{\eta}^\top T(\mathbf{z})). \tag{4.11}$$

The mappings between the mean parameterization $\boldsymbol{\mu}_{\text{opt}}$ and the natural parameterization $\boldsymbol{\eta}$ are formally hard to identify except in special cases [137].

In this manuscript, emergent properties are defined by statistics $f(\mathbf{x};\mathbf{z})$ having a fixed mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$ as in Equation 4.6. The variance constraint is a second moment constraint on $f(\mathbf{x};\mathbf{z})$:

$$\text{Var}_{\mathbf{z},\mathbf{x}}\left[f(\mathbf{x};\mathbf{z})\right] = \mathbb{E}_{\mathbf{z},\mathbf{x}}\left[(f(\mathbf{x};\mathbf{z}) - \boldsymbol{\mu})^2\right]. \tag{4.12}$$

As a general maximum entropy distribution (Equation 4.10), the sufficient statistics vector contains both first and second order moments of $f(\mathbf{x};\mathbf{z})$

$$T(\mathbf{z}) = \begin{bmatrix} \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\mathbf{z})}\left[f(\mathbf{x};\mathbf{z})\right] \\ \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\mathbf{z})}\left[(f(\mathbf{x};\mathbf{z}) - \boldsymbol{\mu})^2\right] \end{bmatrix}, \tag{4.13}$$

which are constrained to the chosen means and variances

$$\boldsymbol{\mu}_{\text{opt}} = \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\sigma}^2 \end{bmatrix}. \tag{4.14}$$

Thus, $\boldsymbol{\mu}_{\text{opt}}$ is used to denote the mean parameter of the maximum entropy distribution defined by the emergent property (all constraints), while $\boldsymbol{\mu}$ is only the mean of $f(\mathbf{x};\mathbf{z})$. The subscript "opt" of $\boldsymbol{\mu}_{\text{opt}}$ is chosen since it contains all of the constraint values to which the EPI optimization algorithm must adhere.

4.4.1.4 Augmented lagrangian optimization

To optimize $q_{\boldsymbol{\theta}}(\mathbf{z})$ in Equation 4.8, the constrained maximum entropy optimization is executed using the augmented lagrangian method. The following objective is minimized:

$$L(\boldsymbol{\theta};\boldsymbol{\eta}_{\text{opt}},c) = -H(q_{\boldsymbol{\theta}}) + \boldsymbol{\eta}_{\text{opt}}^{\top} R(\boldsymbol{\theta}) + \frac{c}{2}||R(\boldsymbol{\theta})||^2 \tag{4.15}$$

where there are average constraint violations

$$R(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z})}\left[T(\mathbf{z}) - \boldsymbol{\mu}_{\text{opt}}\right], \tag{4.16}$$

$\boldsymbol{\eta}_{\text{opt}} \in \mathbb{R}^m$ are the lagrange multipliers where m is the number of total constraints

$$m = |\boldsymbol{\mu}_{\text{opt}}| = |T(\mathbf{z})| = 2|f(\mathbf{x};\mathbf{z})|, \tag{4.17}$$

and c is the penalty coefficient. The mean parameter $\boldsymbol{\mu}_{\text{opt}}$ and sufficient statistics $T(\mathbf{z})$ are determined by the means $\boldsymbol{\mu}$ and variances $\boldsymbol{\sigma}^2$ of the emergent property statistics $f(\mathbf{x};\mathbf{z})$ defined in Equation 4.8. Specifically, $T(\mathbf{z})$ is a concatenation of the first and second moments (Equation 4.13) and $\boldsymbol{\mu}_{\text{opt}}$ is a concatenation of their constraints $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ (Equation 4.14). (Although, note that this algorithm is written for general $T(\mathbf{z})$ and $\boldsymbol{\mu}_{\text{opt}}$ to satisfy the more general class of emergent properties.) The lagrange multipliers $\boldsymbol{\eta}_{\text{opt}}$ are closely related to the natural parameters $\boldsymbol{\eta}$ of exponential families (see Section 4.4.1.6). Weights and biases $\boldsymbol{\theta}$ of the deep probability distribution are optimized according to Equation 4.15 using the Adam optimizer with learning rate 10^{-3} [157].

The gradient with respect to entropy $H(q_{\boldsymbol{\theta}}(\mathbf{z}))$ can be expressed using the reparameterization trick as an expectation of the negative log density of parameter samples $\mathbf{z}$ over the randomness in the parameterless initial distribution $q_0(\mathbf{z}_0)$:

$$H(q_{\boldsymbol{\theta}}(\mathbf{z})) = \int -q_{\boldsymbol{\theta}}(\mathbf{z}) \log(q_{\boldsymbol{\theta}}(\mathbf{z}))d\mathbf{z} = \mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\theta}}} \left[-\log(q_{\boldsymbol{\theta}}(\mathbf{z})) \right] = \mathbb{E}_{\mathbf{z}_0 \sim q_0} \left[-\log(q_{\boldsymbol{\theta}}(g_{\boldsymbol{\theta}}(\mathbf{z}_0))) \right] . \tag{4.18}$$

Thus, the gradient of the entropy of the deep probability distribution can be estimated as an average of gradients with respect to the base distribution $\mathbf{z}_0$:

$$\nabla_{\boldsymbol{\theta}} H(q_{\boldsymbol{\theta}}(\mathbf{z})) = \mathbb{E}_{\mathbf{z}_0 \sim q_0} \left[-\nabla_{\boldsymbol{\theta}} \log(q_{\boldsymbol{\theta}}(g_{\boldsymbol{\theta}}(\mathbf{z}_0))) \right] . \tag{4.19}$$

The gradients of the log density of the deep probability distribution are tractable through the use of normalizing flows (see Section 4.4.1.2).

The full EPI optimization algorithm is detailed in Algorithm 1. The lagrangian parameters $\boldsymbol{\eta}_{\text{opt}}$ are initialized to zero and adapted following each augmented lagrangian epoch, which is a period of optimization with fixed $(\boldsymbol{\eta}_{\text{opt}}, c)$ for a given number of stochastic gradient descent (SGD) iterations. A low value of c is used initially, and conditionally increased after each epoch based on constraint error reduction. The penalty coefficient is updated based on the result of a hypothesis test regarding the reduction in constraint violation. The p-value of $\mathbb{E}[||R(\boldsymbol{\theta}_{k+1})||] > \gamma \mathbb{E}\left[||R(\boldsymbol{\theta}_k)||\right]$ is computed, and c_{k+1} is updated to βc_k with probability $1 - p$. The other update rule is $\boldsymbol{\eta}_{\text{opt},k+1} = \boldsymbol{\eta}_{\text{opt},k} + c_k \frac{1}{n} \sum_{i=1}^n (T(\mathbf{z}^{(i)}) - \boldsymbol{\mu}_{\text{opt}})$ given a batch size n and $\mathbf{z}^{(i)} \sim q_{\boldsymbol{\theta}}(\mathbf{z})$. Throughout the study, $\gamma = 0.25$, while β was chosen to be either 2 or 4. The batch size of EPI also varied according to application.

In general, c and $\boldsymbol{\eta}_{\text{opt}}$ should start at values encouraging entropic growth early in optimization. With each training epoch in which the update rule for c is invoked, the constraint satisfaction terms are increasingly weighted, which generally results in decreased entropy (e.g. see Figure 4.3C). This encourages the discovery of suitable regions of parameter space, and the subsequent refinement of the distribution to produce the emergent property. The momentum parameters of the Adam optimizer are reset at the end of each augmented lagrangian epoch, which proceeds for i_{max}

Algorithm 1: Emergent property inference

```
1  initialize $\boldsymbol{\theta}$ by fitting $q_{\boldsymbol{\theta}}$ to an isotropic gaussian of mean $\boldsymbol{\mu}_{\text{init}}$ and variance $\boldsymbol{\sigma}^2_{\text{init}}$
2  initialize $c_0 > 0$ and $\boldsymbol{\eta}_{\text{opt},0} = \mathbf{0}$.
3  for Augmented lagrangian epoch $k = 1, ..., k_{\max}$ do
4      for SGD iteration $i = 1, ..., i_{\max}$ do
5          Sample $\mathbf{z}_0^{(1)}, ..., \mathbf{z}_0^{(n)} \sim q_0$, get transformed variable $\mathbf{z}^{(j)} = g_{\boldsymbol{\theta}}(\mathbf{z}_0^{(j)})$, $j = 1, ..., n$
6          Update $\boldsymbol{\theta}$ by descending its stochastic gradient (using ADAM optimizer [157]).
              $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \boldsymbol{\eta}_{\text{opt},k}, c) = \frac{1}{n}\sum_{j=1}^{n} \nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\mathbf{z}^{(j)}) + \frac{1}{n}\sum_{j=1}^{n} \nabla_{\boldsymbol{\theta}} \left(T\left(\mathbf{z}^{(j)}\right) - \boldsymbol{\mu}_{\text{opt}}\right) \boldsymbol{\eta}_{\text{opt},k}$
              $+ c_k \frac{2}{n} \sum_{j=1}^{\frac{n}{2}} \nabla_{\boldsymbol{\theta}} \left(T\left(\mathbf{z}^{(j)}\right) - \boldsymbol{\mu}_{\text{opt}}\right) \cdot \frac{2}{n} \sum_{j=\frac{n}{2}+1}^{n} \left(T\left(\mathbf{z}^{(j)}\right) - \boldsymbol{\mu}_{\text{opt}}\right)$
7      end
8      Sample $\mathbf{z}_0^{(1)}, ..., \mathbf{z}_0^{(n)} \sim q_0$, get transformed variable $\mathbf{z}^{(j)} = g_{\boldsymbol{\theta}}(\mathbf{z}_0^{(j)})$, $j = 1, ..., n$
9      Update $\boldsymbol{\eta}_{\text{opt},k+1} = \boldsymbol{\eta}_{\text{opt},k} + c_k \frac{1}{n} \sum_{j=1}^{n} \left(T\left(\mathbf{z}^{(j)}\right) - \boldsymbol{\mu}_{\text{opt}}\right)$.
10     Update $c_{k+1} > c_k$ (see text for detail).
11 end
```

iterations. In this work, we used a maximum number of augmented lagrangian epochs $k_{\max} >= 5$.

Rather than starting optimization from some $\boldsymbol{\theta}$ drawn from a randomized distribution, we found that initializing $q_{\boldsymbol{\theta}}(\mathbf{z})$ to approximate an isotropic gaussian distribution conferred more stable, consistent optimization. The parameters of the gaussian initialization were chosen on an application-specific basis. Throughout the study, we chose isotropic Gaussian initializations with mean $\boldsymbol{\mu}_{\text{init}}$ at the center of the support of the distribution and some variance $\boldsymbol{\sigma}^2_{\text{init}}$, except for one case, where an initialization informed by random search was used (see Section 4.4.2). Deep probability distributions were fit to these gaussian initializations using 10,000 iterations of stochastic gradient descent on the evidence lower bound (as in [124]) with Adam optimizer and a learning rate of 10^{-3}.

To assess whether the EPI distribution $q_{\boldsymbol{\theta}}(\mathbf{z})$ produces the emergent property, we assess whether each individual constraint on the means and variances of $f(\mathbf{x}; \mathbf{z})$ is satisfied. We consider the EPI to have converged when a null hypothesis test of constraint violations $R(\boldsymbol{\theta})_i$ being zero is accepted for all constraints $i \in \{1, ..., m\}$ at a significance threshold $\alpha = 0.05$. This significance threshold is adjusted through Bonferroni correction according to the number of constraints m. The p-values for each constraint are calculated according to a two-tailed nonparametric test, where 200 estimations of the sample mean $R(\boldsymbol{\theta})^i$ are made using N_{test} samples of $\mathbf{z} \sim q_{\boldsymbol{\theta}}(\mathbf{z})$ at the end of the augmented lagrangian epoch. Of all $k_{\max}$ augmented lagrangian epochs, we select the EPI inferred distribution as that which satisfies the convergence criteria and has greatest entropy.

When assessing the suitability of EPI for a particular modeling question, there are some important technical considerations. First and foremost, as in any optimization problem, the defined emergent property should always be appropriately conditioned (constraints should not have wildly different units). Furthermore, if the program is underconstrained (not enough constraints), the distribution grows (in entropy) unstably unless mapped to a finite support. If overconstrained, there is no parameter set producing the emergent property, and EPI optimization will fail (appropriately).

4.4.1.5 Example: 2D LDS

To gain intuition for EPI, consider a two-dimensional linear dynamical system (2D LDS) model (Fig. 4.3A):

$$\tau \frac{d\mathbf{x}}{dt} = A\mathbf{x} \tag{4.20}$$

with

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}. \tag{4.21}$$

To run EPI with the dynamics matrix elements as the free parameters $\mathbf{z} = [a_{1,1}, a_{1,2}, a_{2,1}, a_{2,2}]$ (fixing $\tau = 1$s), the emergent property statistics $f(\mathbf{x}; \mathbf{z})$ were chosen to contain parts of the primary eigenvalue of A, which predicate frequency, $\text{imag}(\lambda_1)$, and the growth/decay, $\text{real}(\lambda_1)$, of the system

$$f(\mathbf{x}; \mathbf{z}) \triangleq \begin{bmatrix} \text{real}(\lambda_1)(\mathbf{x}; \mathbf{z}) \\ \text{imag}(\lambda_1)(\mathbf{x}; \mathbf{z}) \end{bmatrix} \tag{4.22}$$

λ_1 is the eigenvalue of greatest real part when the imaginary component is zero, and alternatively that of positive imaginary component when the eigenvalues are complex conjugate pairs. To learn the distribution of real entries of A that produce a band of oscillating systems around 1Hz, we formalized this emergent property as real(λ_1) having mean zero with variance 0.25^2, and the oscillation frequency $\frac{\text{imag}(\lambda_1)}{2\pi}$ having mean 1Hz with variance 0.1Hz^2:

$$\begin{aligned}\mathcal{X} \; : \; \mathbb{E}_{\mathbf{z},\mathbf{x}}\left[f(\mathbf{x};\mathbf{z})\right] &\triangleq \mathbb{E}_{\mathbf{z},\mathbf{x}}\begin{bmatrix}\text{real}(\lambda_1)(\mathbf{x};\mathbf{z}) \\ \text{imag}(\lambda_1)(\mathbf{x};\mathbf{z})\end{bmatrix} = \begin{bmatrix}0 \\ 2\pi\end{bmatrix} \triangleq \boldsymbol{\mu} \\ \text{Var}_{\mathbf{z},\mathbf{x}}\left[f(\mathbf{x};\mathbf{z})\right] &\triangleq \text{Var}_{\mathbf{z},\mathbf{x}}\begin{bmatrix}\text{real}(\lambda_1)(\mathbf{x};\mathbf{z}) \\ \text{imag}(\lambda_1)(\mathbf{x};\mathbf{z})\end{bmatrix} = \begin{bmatrix}0.25^2 \\ (\frac{\pi}{5})^2\end{bmatrix} \triangleq \boldsymbol{\sigma}^2.\end{aligned} \tag{4.23}$$

To write the emergent property $\mathcal{X}$ in the form required for the augmented lagrangian optimization (Section 4.4.1.4), we concatenate these first and second moment constraints into a vector of sufficient statistics $T(\mathbf{z})$ and constraint values $\boldsymbol{\mu}_{\text{opt}}$.

$$\mathbb{E}_{\mathbf{z}}\left[T(\mathbf{z})\right] \;\triangleq\; \mathbb{E}_{\mathbf{z}}\begin{bmatrix}\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\mathbf{z})}\left[\text{real}(\lambda_1)(\mathbf{x};\mathbf{z})\right] \\ \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\mathbf{z})}\left[\text{imag}(\lambda_1)(\mathbf{x};\mathbf{z})\right] \\ \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\mathbf{z})}\left[(\text{real}(\lambda_1)(\mathbf{x};\mathbf{z})-0)^2\right] \\ \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\mathbf{z})}\left[(\text{imag}(\lambda_1)(\mathbf{x};\mathbf{z})-2\pi)^2\right]\end{bmatrix} = \begin{bmatrix}0 \\ 2\pi \\ 0.25^2 \\ (\frac{\pi}{5})^2\end{bmatrix} \;\triangleq\; \boldsymbol{\mu}_{\text{opt}}. \tag{4.24}$$

From now on in all scientific applications (Sections 4.4.2-5.5.2, we specify how the EPI optimization was setup by specifying $f(\mathbf{x};\mathbf{z})$, $\boldsymbol{\mu}$, and $\boldsymbol{\sigma}^2$.

Unlike the models we presented in the main text, this model admits an analytical form for the mean emergent property statistics given parameter $\mathbf{z}$, since the eigenvalues can be calculated using the quadratic formula:

$$\lambda = \frac{(\frac{a_{1,1}+a_{2,2}}{\tau}) \pm \sqrt{(\frac{a_{1,1}+a_{2,2}}{\tau})^2 + 4(\frac{a_{1,2}a_{2,1}-a_{1,1}a_{2,2}}{\tau})}}{2}. \tag{4.25}$$

We study this example, because the inferred distribution is curved and multimodal, and we can compare the result of EPI to analytically derived contours of the emergent property statistics.

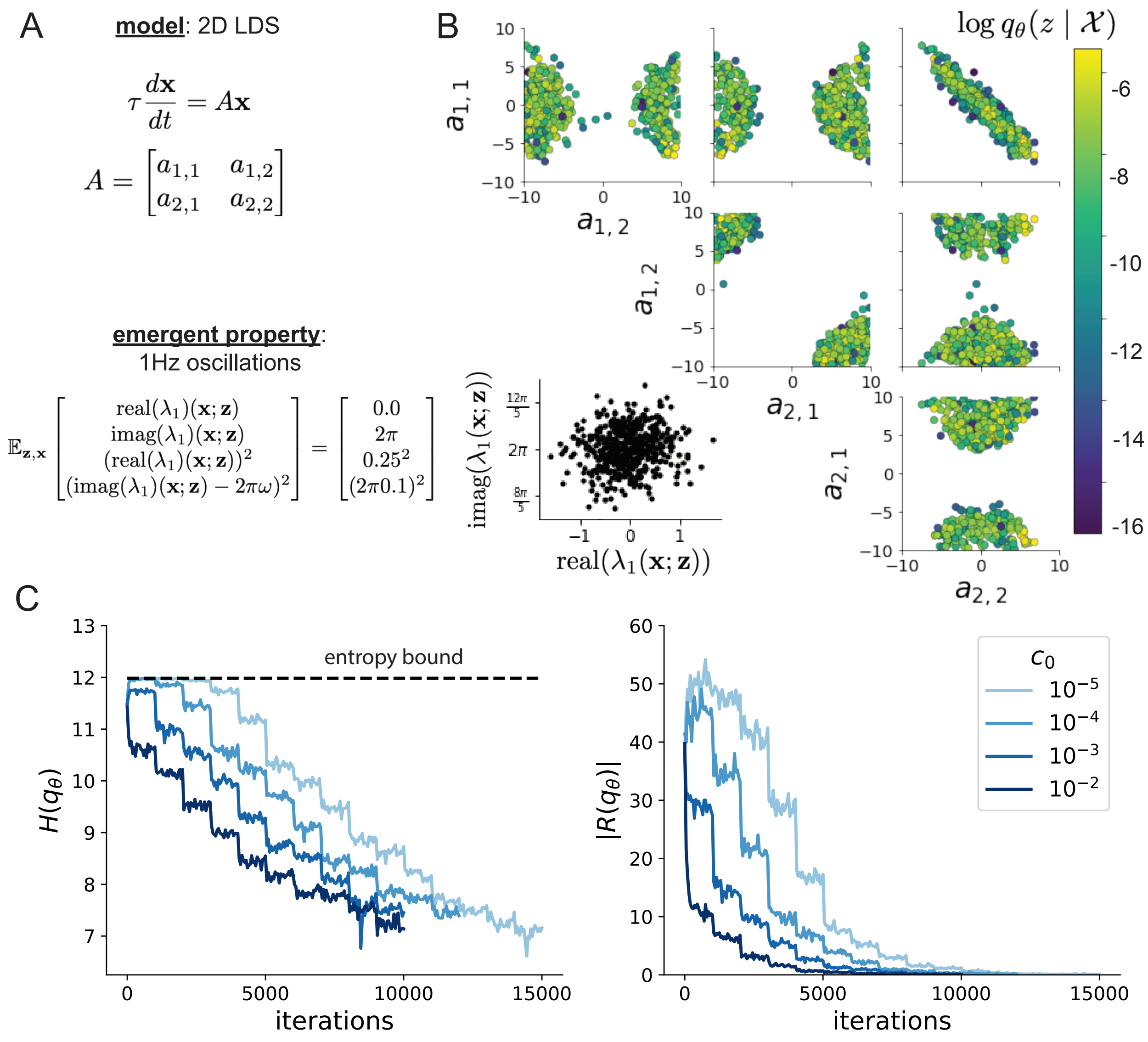


Figure 4.3: **A**. Two-dimensional linear dynamical system model, where real entries of the dynamics matrix A are the parameters. **B**. The EPI distribution for a two-dimensional linear dynamical system with $\tau = 1$ that produces an average of 1Hz oscillations with some small amount of variance. Dashed lines indicate the parameter axes. **C**. Entropy throughout the optimization. At the beginning of each augmented lagrangian epoch ($i_{\max} = 2,000$ iterations), the entropy dipped due to the shifted optimization manifold where emergent property constraint satisfaction is increasingly weighted. **D**. Emergent property moments throughout optimization. At the beginning of each augmented lagrangian epoch, the emergent property moments adjust closer to their constraints.

Despite the simple analytic form of the emergent property statistics, the EPI distribution in this example is not simply determined. Although $\mathbb{E}_{\mathbf{z}}\left[T(\mathbf{z})\right]$ is calculable directly via a closed form function, the distribution $q_{\boldsymbol{\theta}}^{*}(\mathbf{z} \mid \mathcal{X})$ cannot be derived directly. This fact is due to the formally hard problem of the backward mapping: finding the natural parameters $\boldsymbol{\eta}$ from the mean parameters $\boldsymbol{\mu}$ of an exponential family distribution [137]. Instead, we used EPI to approximate this distribution (Fig. 4.3B). We used a real NVP normalizing flow architecture three coupling layers and two-layer neural networks of 50 units per layer, mapped onto a support of $z_i \in [-10, 10]$. (see Section 4.4.1.2).

Even this relatively simple system has nontrivial (though intuitively sensible) structure in the parameter distribution. To validate our method, we analytically derived the contours of the probability density from the emergent property statistics and values. In the $a_{1,1}$-$a_{2,2}$ plane, the black line at $\text{real}(\lambda_1) = \frac{a_{1,1}+a_{2,2}}{2} = 0$, dashed black line at the standard deviation $\text{real}(\lambda_1) = \frac{a_{1,1}+a_{2,2}}{2} \pm 0.25$, and the dashed gray line at twice the standard deviation $\text{real}(\lambda_1) = \frac{a_{1,1}+a_{2,2}}{2} \pm 0.5$ follow the contour of probability density of the samples (Fig. 4.4A). The distribution precisely reflects the desired statistical constraints and model degeneracy in the sum of $a_{1,1}$ and $a_{2,2}$. Intuitively, the parameters equivalent with respect to emergent property statistic $\text{real}(\lambda_1)$ have similar log densities.

To explain the bimodality of the EPI distribution, we examined the imaginary component of λ_1. When $\text{real}(\lambda_1) = a_{1,1} + a_{2,2} = 0$ (which is the case on average in $\mathcal{X}$), we have

$$\text{imag}(\lambda_1) = \begin{cases} \sqrt{\frac{a_{1,1}a_{2,2}-a_{1,2}a_{2,1}}{\tau}}, & \text{if } a_{1,1}a_{2,2} < a_{1,2}a_{2,1} \\ 0 & \text{otherwise} \end{cases}. \tag{4.26}$$

In Figure 4.4B, we plot the contours of $\text{imag}(\lambda_1)$ where $a_{1,1}a_{2,2}$ is fixed to 0 at one standard deviation ($\frac{\pi}{5}$, black dashed) and two standard deviations ($\frac{2\pi}{5}$, gray dashed) from the mean of 2π. This validates the curved multimodal structure of the inferred distribution learned through EPI. Subtler combinations of model and emergent property will have more complexity, further motivating the use of EPI for understanding these systems. As we expect, the distribution results in samples of

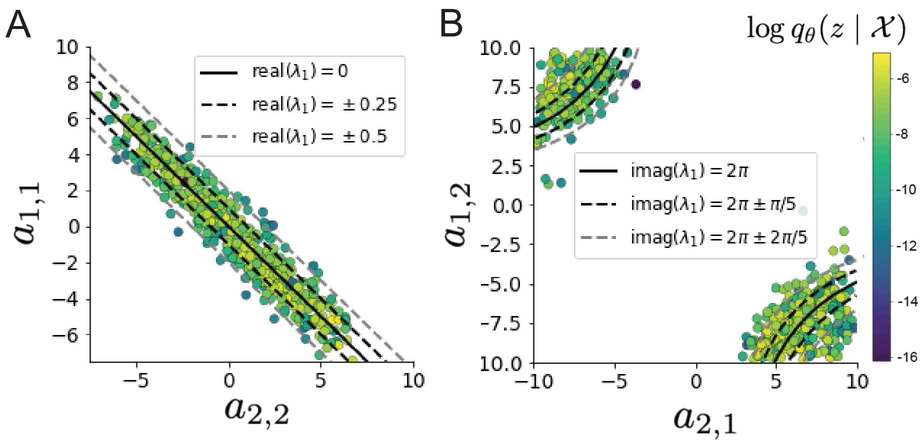


Figure 4.4: **A**. Probability contours in the $a_{1,1}$-$a_{2,2}$ plane were derived from the relationship to emergent property statistic of growth/decay factor real(λ_1). **B**. Probability contours in the $a_{1,2}$-$a_{2,1}$ plane were derived from the emergent property statistic of oscillation frequency 2πimag(λ_1).

two-dimensional linear systems oscillating near 1Hz (Fig. 4.5).

4.4.1.6 EPI as variational inference

In variational inference, a posterior approximation $q^*_{\boldsymbol{\theta}}$ is chosen from within some variational family Q to be as close as possible to the posterior under the KL divergence criteria

$$q^*_{\boldsymbol{\theta}}(\mathbf{z}) = \underset{q_{\boldsymbol{\theta}} \in Q}{\text{argmin}}\, KL(q_{\boldsymbol{\theta}}(\mathbf{z}) \mid\mid p(\mathbf{z} \mid \mathbf{x})). \tag{4.27}$$

This KL divergence can be written in terms of entropy of the variational approximation:

$$KL(q_{\boldsymbol{\theta}}(\mathbf{z}) \mid\mid p(\mathbf{z} \mid \mathbf{x})) = \mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\theta}}} \left[\log(q_{\boldsymbol{\theta}}(\mathbf{z}))\right] - \mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\theta}}} \left[\log(p(\mathbf{z} \mid \mathbf{x}))\right] \tag{4.28}$$

$$= -H(q_{\boldsymbol{\theta}}) - \mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\theta}}} \left[\log(p(\mathbf{x} \mid \mathbf{z})) + \log(p(\mathbf{z})) - \log(p(\mathbf{x}))\right] \tag{4.29}$$

Since the marginal distribution of the data $p(\mathbf{x})$ (or “evidence") is independent of $\boldsymbol{\theta}$, variational inference is executed by optimizing the remaining expression. This is usually framed as maximizing

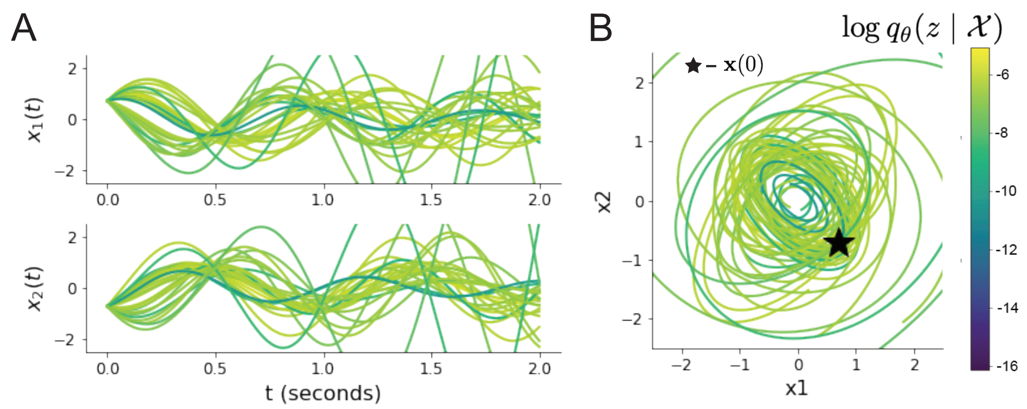


Figure 4.5: Sampled dynamical systems $\mathbf{z} \sim q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathcal{X})$ and their simulated activity from $\mathbf{x}(t = 0) = [\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}]$ colored by log probability. **A**. Each dimension of the simulated trajectories throughout time. **B**. The simulated trajectories in phase space.

the evidence lower bound (ELBO)

$$\underset{q_{\boldsymbol{\theta}} \in Q}{\text{argmin}}\, KL(q_{\boldsymbol{\theta}} \mid\mid p(\mathbf{z} \mid \mathbf{x})) = \underset{q_{\boldsymbol{\theta}} \in Q}{\text{argmax}}\, H(q_{\boldsymbol{\theta}}) + \mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\theta}}} \left[\log(p(\mathbf{x} \mid \mathbf{z})) + \log(p(\mathbf{z}))\right]. \tag{4.30}$$

Now, we will show how the maximum entropy problem of EPI is equivalent to variational inference. In general, a maximum entropy problem (as in Equation 4.10) has an equivalent lagrange dual form:

$$\begin{aligned} &\underset{q \in Q}{\text{argmax}}\, H(q(\mathbf{z})) \qquad \Longleftrightarrow \qquad \underset{q \in Q}{\text{argmax}}\, H(q(\mathbf{z})) + \boldsymbol{\eta}^{*\top} \mathbb{E}_{\mathbf{z} \sim q} \left[T(\mathbf{z})\right], \\ &\text{s.t. } \mathbb{E}_{\mathbf{z} \sim q} \left[T(\mathbf{z})\right] = \mathbf{0} \end{aligned} \tag{4.31}$$

with lagrange multipliers $\boldsymbol{\eta}^*$. By moving the lagrange multipliers within the expectation

$$q^* = \underset{q \in Q}{\text{argmax}}\, H(q(\mathbf{z})) + \mathbb{E}_{\mathbf{z} \sim q} \left[\boldsymbol{\eta}^{*\top} T(\mathbf{z})\right], \tag{4.32}$$

inserting a $\log \exp(\cdot)$ within the expectation,

$$q^* = \underset{q \in Q}{\text{argmax}}\, H(q(\mathbf{z})) + \mathbb{E}_{\mathbf{z} \sim q} \left[\log \exp\left(\boldsymbol{\eta}^{*\top} T(\mathbf{z})\right)\right], \tag{4.33}$$

and finally choosing $T(\cdot)$ to be likelihood averaged statistics as in EPI

$$q^* = \underset{q \in Q}{\operatorname{argmax}} \, H(q(\mathbf{z})) + \mathbb{E}_{\mathbf{z} \sim q} \left[\log \exp \left(\boldsymbol{\eta}^{*\top} \begin{bmatrix} \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})} \left[\phi_1(\mathbf{x}; \mathbf{z}) \right] \\ \ldots \\ \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})} \left[\phi_m(\mathbf{x}; \mathbf{z}) \right] \end{bmatrix} \right) \right], \tag{4.34}$$

we can compare directly to the objective used in variational inference (Equation 4.30). We see that EPI is exactly variational inference with an exponential family likelihood defined by sufficient statistics $T(\mathbf{z}) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})} \left[\phi(\mathbf{x}; \mathbf{z}) \right]$, and where the natural parameter $\boldsymbol{\eta}^*$ is predicated by the choice of mean parameter $\boldsymbol{\mu}_{\text{opt}}$. Equation 4.34 implies that EPI uses an improper (or uniform) prior, which is easily changed.

This derivation of the equivalence between EPI and variational inference emphasizes why defining a statistical inference program by its mean parameterization $\boldsymbol{\mu}_{\text{opt}}$ is so useful. With EPI, one can clearly define the emergent property $\mathcal{X}$ that the model of interest should produce through intuitive selection of $\boldsymbol{\mu}_{\text{opt}}$ for a given $T(\mathbf{z})$. Alternatively, figuring out the correct natural parameters $\boldsymbol{\eta}^*$ for the same $T(\mathbf{z})$ that produces $\mathcal{X}$ is a formally hard problem (see Section 4.4.1.3).

4.4.2 Stomatogastric ganglion

In Section 4.2.1 and 4.2.2, we used EPI to infer conductance parameters in a model of the stomatogastric ganglion (STG) [41]. This 5-neuron circuit model represents two subcircuits: that generating the pyloric rhythm (fast population) and that generating the gastric mill rhythm (slow population). The additional neuron (the IC neuron of the STG) receives inhibitory synaptic input from both subcircuits, and can couple to either rhythm dependent on modulatory conditions. There is also a parametric regime in which this neuron fires at an intermediate frequency between that of the fast and slow populations [41], which we infer with EPI as a motivational example. This model is not to be confused with an STG subcircuit model of the pyloric rhythm [203], which has been statistically inferred in other studies [173, 190].

4.4.2.1 STG model

We analyze how the parameters $\mathbf{z} = [g_{\text{el}}, g_{\text{synA}}]$ govern the emergent phenomena of intermediate hub frequency in a model of the stomatogastric ganglion (STG) [41] shown in Figure 4.1A with activity $\mathbf{x} = [x_{\text{f1}}, x_{\text{f2}}, x_{\text{hub}}, x_{\text{s1}}, x_{\text{s2}}]$, using the same hyperparameter choices as Gutierrez et al. Each neuron's membrane potential $x_\alpha(t)$ for $\alpha \in \{\text{f1}, \text{f2}, \text{hub}, \text{s1}, \text{s2}\}$ is the solution of the following stochastic differential equation:

$$C_m \frac{dx_\alpha}{dt} = -\left[h_{leak}(\mathbf{x};\mathbf{z}) + h_{Ca}(\mathbf{x};\mathbf{z}) + h_K(\mathbf{x};\mathbf{z}) + h_{hyp}(\mathbf{x};\mathbf{z}) + h_{elec}(\mathbf{x};\mathbf{z}) + h_{syn}(\mathbf{x};\mathbf{z})\right] + dB. \tag{4.35}$$

The input current of each neuron is the sum of the leak, calcium, potassium, hyperpolarization, electrical and synaptic currents. Each current component is a function of all membrane potentials and the conductance parameters $\mathbf{z}$. Finally, we include gaussian noise dB to the model of Gutierrez et al. so that the model stochastic, although this is not required by EPI.

The capacitance of the cell membrane was set to $C_m = 1nF$. Specifically, the currents are the difference in the neuron's membrane potential and that current type's reversal potential multiplied by a conductance:

$$h_{leak}(\mathbf{x};\mathbf{z}) = g_{leak}(x_\alpha - V_{leak}) \tag{4.36}$$

$$h_{elec}(\mathbf{x};\mathbf{z}) = g_{\text{el}}(x_\alpha^{post} - x_\alpha^{pre}) \tag{4.37}$$

$$h_{syn}(\mathbf{x};\mathbf{z}) = g_{syn} S_\infty^{pre}(x_\alpha^{post} - V_{syn}) \tag{4.38}$$

$$h_{Ca}(\mathbf{x};\mathbf{z}) = g_{Ca} M_\infty (x_\alpha - V_{Ca}) \tag{4.39}$$

$$h_K(\mathbf{x};\mathbf{z}) = g_K N (x_\alpha - V_K) \tag{4.40}$$

$$h_{hyp}(\mathbf{x};\mathbf{z}) = g_h H (x_\alpha - V_{hyp}). \tag{4.41}$$

The reversal potentials were set to $V_{leak} = -40mV$, $V_{Ca} = 100mV$, $V_K = -80mV$, $V_{hyp} = -20mV$, and $V_{syn} = -75mV$. The other conductance parameters were fixed to $g_{leak} = 1 \times 10^{-4} \mu S$. g_{Ca},

g_K, and g_{hyp} had different values based on fast, intermediate (hub) or slow neuron. The fast conductances had values $g_{Ca} = 1.9 \times 10^{-2}$, $g_K = 3.9 \times 10^{-2}$, and $g_{hyp} = 2.5 \times 10^{-2}$. The intermediate conductances had values $g_{Ca} = 1.7 \times 10^{-2}$, $g_K = 1.9 \times 10^{-2}$, and $g_{hyp} = 8.0 \times 10^{-3}$. Finally, the slow conductances had values $g_{Ca} = 8.5\times10^{-3}$, $g_K = 1.5\times10^{-2}$, and $g_{hyp} = 1.0\times10^{-2}$.

Furthermore, the Calcium, Potassium, and hyperpolarization channels have time-dependent gating dynamics dependent on steady-state gating variables M_∞, N_∞ and H_∞, respectively:

$$M_\infty = 0.5\left(1 + \tanh\left(\frac{x_\alpha - v_1}{v_2}\right)\right) \tag{4.42}$$

$$\frac{dN}{dt} = \lambda_N(N_\infty - N) \tag{4.43}$$

$$N_\infty = 0.5\left(1 + \tanh\left(\frac{x_\alpha - v_3}{v_4}\right)\right) \tag{4.44}$$

$$\lambda_N = \phi_N \cosh\left(\frac{x_\alpha - v_3}{2v_4}\right) \tag{4.45}$$

$$\frac{dH}{dt} = \frac{(H_\infty - H)}{\tau_h} \tag{4.46}$$

$$H_\infty = \frac{1}{1 + \exp\left(\frac{x_\alpha + v_5}{v_6}\right)} \tag{4.47}$$

$$\tau_h = 272 - \left(\frac{-1499}{1 + \exp\left(\frac{-x_\alpha + v_7}{v_8}\right)}\right). \tag{4.48}$$

where we set $v_1 = 0mV$, $v_2 = 20mV$, $v_3 = 0mV$, $v_4 = 15mV$, $v_5 = 78.3mV$, $v_6 = 10.5mV$, $v_7 = -42.2mV$, $v_8 = 87.3mV$, $v_9 = 5mV$, and $v_{th} = -25mV$.

Finally, there is a synaptic gating variable as well:

$$S_\infty = \frac{1}{1 + \exp\left(\frac{v_{th} - x_\alpha}{v_9}\right)}. \tag{4.49}$$

When the dynamic gating variables are considered, this is actually a 15-dimensional nonlinear dynamical system. The gaussian noise $d\mathbf{B}$ has variance $(1 \times 10^{-12})^2$ A^2, and introduces variability in frequency at each parameterization $\mathbf{z}$.

4.4.2.2 Hub frequency calculation

In order to measure the frequency of the hub neuron during EPI, the STG model was simulated for $T = 300$ time steps of $dt = 25$ms. The chosen dt and T were the most computationally convenient choices yielding accurate frequency measurement. We used a basis of complex exponentials with frequencies from 0.0-1.0 Hz at 0.01Hz resolution to measure frequency from simulated time series

$$\Phi = [0.0, 0.01, ..., 1.0]^\top .. \tag{4.50}$$

To measure spiking frequency, we processed simulated membrane potentials with a relu (spike extraction) and low-pass filter with averaging window of size 20, then took the frequency with the maximum absolute value of the complex exponential basis coefficients of the processed time-series. The first 20 temporal samples of the simulation are ignored to account for initial transients.

To differentiate through the maximum frequency identification, we used a soft-argmax Let $X_\alpha \in C^{|\Phi|}$ be the complex exponential filter bank dot products with the signal $x_\alpha \in \mathbb{R}^N$, where $\alpha \in \{\text{f1}, \text{f2}, \text{hub}, \text{s1}, \text{s2}\}$. The soft-argmax is then calculated using temperature parameter $\beta_\psi = 100$

$$\psi_\alpha = \text{softmax}(\beta_\psi |X_\alpha|) \odot i, \tag{4.51}$$

where $i = [0, 1, ..., 100]$. Thus, ψ_α is a dot product between a vector of positive frequency intensities that sum to one, and the corresponding indices to those frequencies. By increasing the temperature parameter β_ψ, the first argument of the dot product more closely approximates a one-hot vector. The frequency is then calculated as

$$\omega_\alpha = 0.01 \psi_\alpha \text{Hz}. \tag{4.52}$$

Intermediate hub frequency, like all other emergent properties in this work, is defined by the

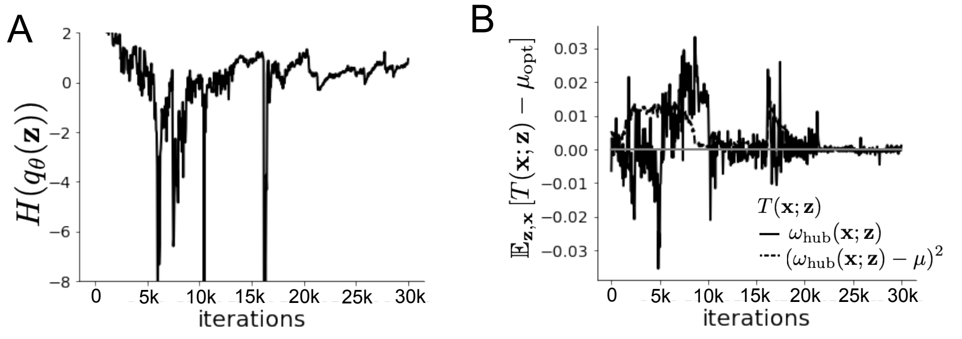


Figure 4.6: EPI optimization of the STG model producing network syncing. **A**. Entropy throughout optimization. **B**. The emergent property statistic means and variances converge to their constraints at 25,000 iterations following the fifth augmented lagrangian epoch.

mean and variance of the emergent property statistics. In this case, we have one statistic, hub neuron frequency, where the mean was chosen to be 0.55Hz,(Equation 4.2) and variance was chosen to be 0.025^2 Hz^2 (Equation 4.3).

4.4.2.3 EPI details for the STG model

EPI was run for the STG model using

$$f(\mathbf{x};\mathbf{z}) = \omega_{\text{hub}}(\mathbf{x};\mathbf{z}), \tag{4.53}$$

$$\boldsymbol{\mu} = \begin{bmatrix} 0.55 \end{bmatrix}, \tag{4.54}$$

and

$$\boldsymbol{\sigma}^2 = \begin{bmatrix} 0.025^2 \end{bmatrix} \tag{4.55}$$

(see Sections 4.4.1.3-4.4.1.4, and example in Section 4.4.1.5). Throughout optimization, the augmented lagrangian parameters η and c, were updated after each epoch of $i_{\text{max}} = 5,000$ iterations (see Section 4.4.1.4). The optimization converged after five epochs (Fig. 4.6).

For EPI in Fig 4.1E, we used a real NVP architecture with three coupling layers and two-layer neural networks of 25 units per layer. The normalizing flow architecture mapped $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, I)$ to a support of $\mathbf{z} = [g_{\text{el}}, g_{\text{synA}}] \in [4, 8] \times [0.01, 4]$, initialized to a gaussian approximation of samples returned by a preliminary ABC search. We did not include $g_{\text{synA}} < 0.01$, for numerical stability. EPI optimization was run using 5 different random seeds for architecture initialization $\boldsymbol{\theta}$ with an augmented lagrangian coefficient of $c_0 = 10^5$, $\beta = 2$, a batch size $n = 400$, and we simulated one $\mathbf{x}^{(i)}$ per $\mathbf{z}^{(i)}$. The architecture converged with criteria $N_{\text{test}} = 100$.

4.4.2.4 Hessian sensitivity vectors

To quantify the second-order structure of the EPI distribution, we evaluated the Hessian of the log probability $\frac{\partial^2 \log q(\mathbf{z} \mid \mathcal{X})}{\partial \mathbf{z}\mathbf{z}^\top}$. The eigenvector of this Hessian with most negative eigenvalue is defined as the sensitivity dimension $\mathbf{v}_1$, and all subsequent eigenvectors are ordered by increasing eigenvalue. These eigenvalues are quantifications of how fast the emergent property deteriorates via the parameter combination of their associated eigenvector. In Figure 4.1D, the sensitivity dimension v_1 (solid) and the second eigenvector of the Hessian v_2 (dashed) are shown evaluated at the mode of the distribution. Since the Hessian eigenvectors have sign degeneracy, the visualized directions in 2-D parameter space were chosen to have positive g_{synA}. The length of the arrows is inversely proportional to the square root of the absolute value of their eigenvalues $\lambda_1 = -10.7$ and $\lambda_2 = -3.22$. For the same magnitude perturbation away from the mode, intermediate hub frequency only diminishes along the sensitivity dimension $\mathbf{v}_1$ (Fig. 4.1E-F).

4.4.3 Scaling EPI for stable amplification in RNNs

4.4.3.1 Rank-2 RNN model

We examined the scaling properties of EPI by learning connectivities of RNNs of increasing size that exhibit stable amplification. Rank-2 RNN connectivity was modeled as $W = UV^\top$, where $U = \begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_2 \end{bmatrix} + g\chi^{(W)}$, $V = \begin{bmatrix} \mathbf{V}_1 & \mathbf{V}_2 \end{bmatrix} + g\chi^{(V)}$, and $\chi^{(W)}_{i,j}, \chi^{(V)}_{i,j} \sim \mathcal{N}(0, 1)$. This RNN model has dynamics

$$\tau \dot{\mathbf{x}} = -\mathbf{x} + W\mathbf{x}. \tag{4.56}$$

In this analysis, we inferred connectivity parameterizations $\mathbf{z} = \left[\mathbf{U}_1^\top, \mathbf{U}_2^\top, \mathbf{V}_1^\top, \mathbf{V}_2^\top\right]^\top \in [-1, 1]^{(4N)}$ that produced stable amplification using EPI, SMC-ABC [52], and SNPE [190] (see Section Related Methods).

4.4.3.2 Stable amplification

For this RNN model to be stable, all real eigenvalues of W must be less than 1: $\text{real}(\lambda_1) < 1$, where λ_1 denotes the greatest real eigenvalue of W. For a stable RNN to amplify at least one input pattern, the symmetric connectivity $W^s = \frac{W+W\top}{2}$ must have an eigenvalue greater than 1: $\lambda_1^s > 1$, where λ^s is the maximum eigenvalue of W^s. These two conditions are necessary and sufficient for stable amplification in RNNs [201].

4.4.3.3 EPI details for RNNs

We defined the emergent property of stable amplification with means of these eigenvalues (0.5 and 1.5, respectively) that satisfy these conditions. To complete the emergent property definition, we chose variances (0.25^2) about those means such that samples rarely violate the eigenvalue constraints. To write the emergent property of Equation 4.5 in terms of the EPI optimization, we have

$$f(\mathbf{x}; \mathbf{z}) = \begin{bmatrix} \text{real}(\lambda_1)(\mathbf{x}; \mathbf{z}) \\ \lambda_1^s(\mathbf{x}; \mathbf{z}) \end{bmatrix}, \tag{4.57}$$

$$\boldsymbol{\mu} = \begin{bmatrix} 0.5 \\ 1.5 \end{bmatrix}, \tag{4.58}$$

and

$$\boldsymbol{\sigma}^2 = \begin{bmatrix} 0.25^2 \\ 0.25^2 \end{bmatrix} \tag{4.59}$$

(see Sections 4.4.1.3-4.4.1.4, and example in Section 4.4.1.5). Gradients of maximum eigenvalues of Hermitian matrices like W^s are available with modern automatic differentiation tools. To differentiate through the $\text{real}(\lambda_1)$, we solved the following equation for eigenvalues of rank-2 matrices using the rank reduced matrix $W^r = V^\top U$

$$\lambda_\pm = \frac{\text{Tr}(W^r) \pm \sqrt{\text{Tr}(W^r)^2 - 4\text{Det}(W^r)}}{2}. \tag{4.60}$$

For EPI in Fig. 4.2, we used a real NVP architecture with three coupling layers of affine transformations parameterized by two-layer neural networks of 100 units per layer. The initial distribution was a standard isotropic gaussian $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, I)$ mapped to the support of $\mathbf{z}_i \in [-1, 1]$. We used an augmented lagrangian coefficient of $c_0 = 10^3$, a batch size $n = 200$, $\beta = 4$, and we simulated one $\mathbf{W}^{(i)}$ per $\mathbf{z}^{(i)}$. We chose to use $i_{\max} = 500$ iterations per augmented lagrangian epoch and emergent property constraint convergence was evaluated at $N_{\text{test}} = 200$ (Fig. 4.2B blue line, and Fig. 4.2C-D blue). It was fastest to initialize the EPI distribution on a Tesla V100 GPU, and then subsequently optimize it on a CPU with 32 cores. EPI timing measurements accounted for this initialization period.

4.4.3.4 Methodological comparison

We compared EPI to two alternative simulation-based inference techniques, since the likelihood of these eigenvalues given $\mathbf{z}$ is not available. Approximate bayesian computation (ABC) [51] is a rejection sampling technique for obtaining sets of parameters $\mathbf{z}$ that produce activity $\mathbf{x}$ close to some observed data $\mathbf{x}_0$. Sequential Monte Carlo approximate bayesian computation (SMC-ABC) is the state-of-the-art ABC method, which leverages SMC techniques to improve sampling speed. We ran SMC-ABC with the pyABC package [224] to infer RNNs with stable amplification: connectivities having eigenvalues within an ϵ-defined l-2 distance of

$$\mathbf{x}_0 = \begin{bmatrix} \text{real}(\lambda_1) \\ \lambda_1^s \end{bmatrix} = \begin{bmatrix} 0.5 \\ 1.5 \end{bmatrix}. \tag{4.61}$$

SMC-ABC was run with a uniform prior over $\mathbf{z} \in [-1, 1]^{(4N)}$, a population size of 1,000 particles with simulations parallelized over 32 cores, and a multivariate normal transition model.

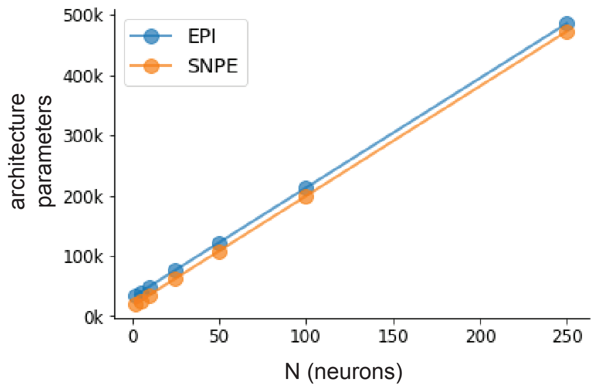


Figure 4.7: Number of parameters in deep probability distribution architectures of EPI (blue) and SNPE (orange) by RNN size (N).

SNPE, the next approach in our comparison, is far more similar to EPI. Like EPI, SNPE treats parameters in mechanistic models with deep probability distributions, yet the two learning algorithms are categorically different. SNPE uses a two-network architecture to approximate the posterior distribution of the model conditioned on observed data $\mathbf{x}_0$. The amortizing network maps observations $\mathbf{x}_i$ to the parameters of the deep probability distribution. The weights and biases of the parameter network are optimized by sequentially augmenting the training data with additional pairs $(\mathbf{z}_i, \mathbf{x}_i)$ based on the most recent posterior approximation. This sequential procedure is important to get training data $\mathbf{z}_i$ to be closer to the true posterior, and $\mathbf{x}_i$ to be closer to the observed data. For the deep probability distribution architecture, we chose a masked autoregressive flow with affine couplings (the default choice), three transforms, 50 hidden units, and a normalizing flow mapping to the support as in EPI. This architectural choice closely tracked the size of the architecture used by EPI (Fig. 4.7). As in SMC-ABC, we ran SNPE with $\mathbf{x}_0 = \mu$. All SNPE optimizations were run for a limit of 1.5 days, or until two consecutive rounds resulted in a validation log probability lower than the maximum observed for that random seed. It was always faster to run SNPE on a CPU with 32 cores rather than on a Tesla V100 GPU.

To compare the efficiency of these algorithms for inferring RNN connectivity distributions producing stable amplification, we develop a convergence criteria that can be used across methods. While EPI has its own hypothesis testing convergence criteria for the emergent property, it would not make sense to use this criteria on SNPE and SMC-ABC which do not constrain the means and variances of their predictions. Instead, we consider EPI and SNPE to have converged after completing its most recent optimization epoch (EPI) or round (SNPE) in which the distance $|\mathbb{E}_{\mathbf{z},\mathbf{x}}\left[f(\mathbf{x};\mathbf{z})\right] - \boldsymbol{\mu}|_2$ is less than 0.5. We consider SMC-ABC to have converged once the population produces samples within the $\epsilon = 0.5$ ball ensuring stable amplification.

When assessing the scalability of SNPE, it is important to check that alternative hyperparameterizations could not yield better performance. Key hyperparameters of the SNPE optimization are the number of simulations per round n_{round}, the number of atoms used in the atomic proposals of the SNPE-C algorithm [225], and the batch size n. To match EPI, we used a batch size of $n = 200$ for $N <= 25$, however we found $n = 1,000$ to be helpful for SNPE in higher dimensions. While $n_{\text{round}} = 1,000$ yielded SNPE convergence for $N <= 25$, we found that a substantial increase to $n_{\text{round}} = 25,000$ yielded more consistent convergence at $N = 50$ (Fig. 4.8A). By increasing n_{round}, we also necessarily increase the duration of each round. At $N = 100$, we tried two hyperparameter modifications. As suggested in [225], we increased n_{atom} by an order of magnitude to improve gradient quality, but this had little effect on the optimization (much overlap between same random seeds) (Fig. 4.8B). Finally, we increased n_{round} by an order of magnitude, which yielded convergence in one case, but no others. We found no way to improve the convergence rate of SNPE without making more aggressive hyperparameter choices requiring high numbers of simulations. In Figure 4.2C-D, we show samples from the random seed resulting in emergent property convergence at greatest entropy (EPI), the random seed resulting in greatest validation log probability (SNPE), and the result of all converged random seeds (SMC).

4.4.3.5 Effect of RNN parameters on EPI and SNPE inferred distributions

To clarify the difference in objectives of EPI and SNPE, we show their results on RNN models with different numbers of neurons N and random strength g. The parameters inferred by EPI consistently produces the same mean and variance of real(λ_1) and λ_1^s, while those inferred by

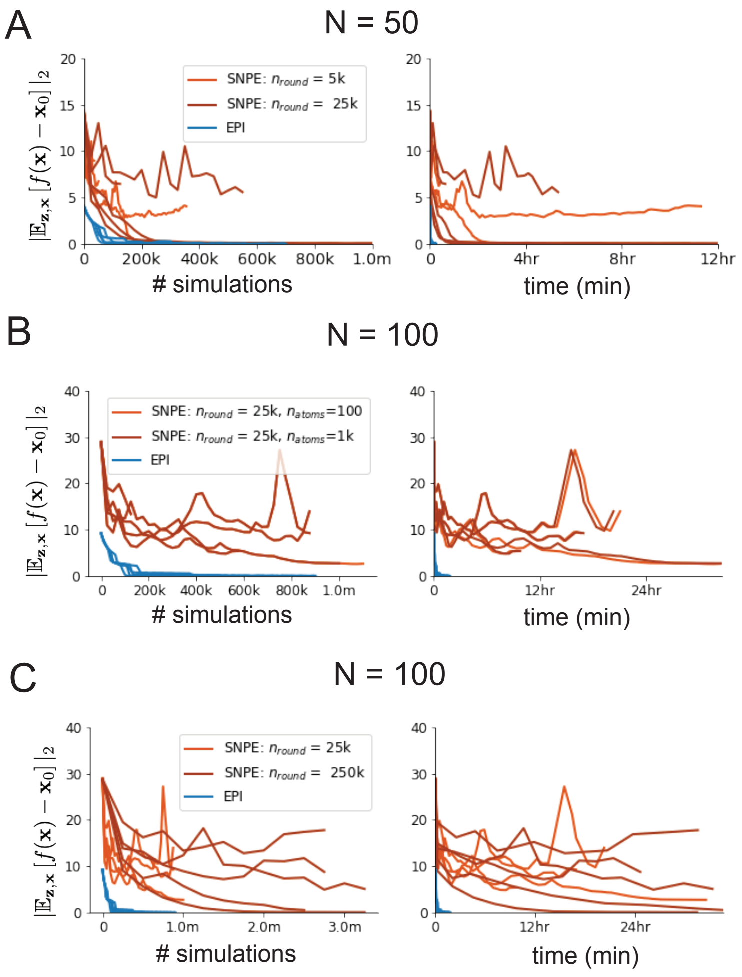


Figure 4.8: SNPE convergence was enabled by increasing n_{round}, not n_{atom}. **A**. Difference of mean predictions $\mathbf{x}_0$ throughout optimization at $N = 50$ with by simulation count (left) and wall time (right) of SNPE with $n_{\text{round}} = 5,000$ (light orange), SNPE with $n_{\text{round}} = 25,000$ (dark orange), and EPI (blue). Each line shows an individual random seed. **B**. Same conventions as A at $N = 100$ of SNPE with $n_{\text{atom}} = 100$ (light orange) and $n_{\text{atom}} = 1,000$ (dark orange). **C**. Same conventions as A at $N = 100$ of SNPE with $n_{\text{round}} = 25,000$ (light orange) and $n_{\text{round}} = 250,000$ (dark orange).

SNPE change according to the model definition (Fig. 4.9A). For $N = 2$ and $g = 0.01$, the SNPE posterior has greater concentration in eigenvalues around $\mathbf{x}_0$ than at $g = 0.1$, where the model has greater randomness (Fig. 4.9B top, orange). At both levels of g when $N = 2$, the posterior of SNPE has lower entropy than EPI at convergence (Fig. 4.9B top). However at $N = 10$, SNPE results in a predictive distribution of more widely dispersed eigenvalues (Fig. 4.9A bottom), and an inferred posterior with greater entropy than EPI (Fig. 4.9B bottom). We highlight these differences not to focus on an insightful trend, but to emphasize that these methods optimize different objectives with different implications.

Note that SNPE converges when it's validation log probability has saturated after several rounds of optimization (Fig. 4.9C), and that EPI converges after several epochs of its own optimization to enforce the emergent property constraints (Fig. 4.9D blue). Importantly, as SNPE optimizes its posterior approximation, the predictive means change, and at convergence may be different than $\mathbf{x}_0$ (Fig. 4.9D orange, left). It is sensible to assume that predictions of a well-approximated SNPE posterior should closely reflect the data on average (especially given a uniform prior and a low degree of stochasticity), however this is not a given. Furthermore, no aspect of the SNPE optimization controls the variance of the predictions (Fig. 4.9D orange, right).

4.5 Deep inference and the exponential family

In Chapter 1 Section 1.3 we introduced the topic of deep generative modeling, and inference with normalizing flows (or deep probability distributions) for the flexible approximation of posterior distributions. In Chapter 3, we presented our novel approach for deep inference of intractable exponential family models using exponential family networks (EFNs). In this chapter, we have presented an additional novel method for deep inference of neural circuit models called emergent property inference (EPI). Furthermore, we compared EPI to an alternative simulation-based inference technique called sequential neural posterior estimation (SNPE) [190], which employs the same two-network architecture as an EFN. Here, we explain mathematical connections and practical similarities between these deep inference techniques through the lens of exponential family

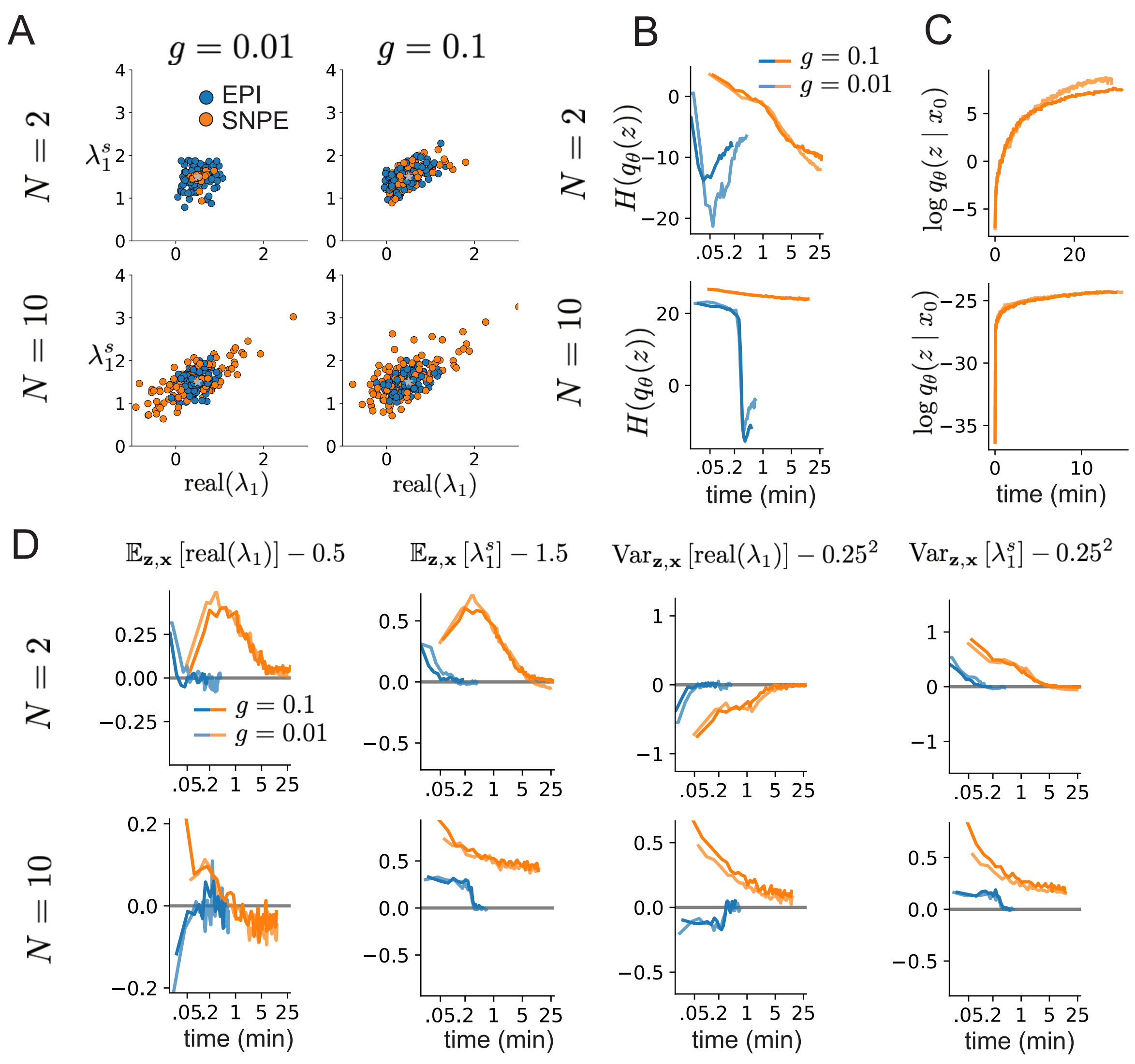


Figure 4.9: Model characteristics affect predictions of posteriors inferred by SNPE, while predictions of parameters inferred by EPI remain fixed. **A**. Predictive distribution of EPI (blue) and SNPE (orange) inferred connectivity of RNNs exhibiting stable amplification with $N = 2$ (top), $N = 10$ (bottom), $g = 0.01$ (left), and $g = 0.1$ (right). **B**. Entropy of parameter distribution approximations throughout optimization with $N = 2$ (top), $N = 10$ (bottom), $g = 0.1$ (dark shade), and $g = 0.01$ (light shade). **C**. Validation log probabilities throughout SNPE optimization. Same conventions as B. **D**. Adherence to EPI constraints. Same conventions as B.

distributions.

4.5.1 Maximum entropy and the exponential family

To formalize the relationship between EPI and EFN, we recognize the equivalence between maximum entropy distributions and their representations as exponential families [137]. Consider a maximum entropy distribution $p^*(\mathbf{z})$ within family $\mathcal{P}$ of form

$$p^*(\mathbf{z}) = \underset{p \in \mathcal{P}}{\text{argmax}}\, H(p(\mathbf{z})) \quad \text{s.t. } \mathbb{E}_{\mathbf{z} \sim p}\left[T(\mathbf{z})\right] = \boldsymbol{\mu}, \tag{4.62}$$

where $H(\cdot)$ is entropy, $T(\cdot)$ is a vector of statistics, and $\boldsymbol{\mu}$ is the mean parameter. The maximum entropy distribution has a solution in the exponential family with parameter $\boldsymbol{\eta}$

$$p^*(\mathbf{z}) \propto \exp(\boldsymbol{\eta}^\top T(\mathbf{z})). \tag{4.63}$$

Thus, every maximum entropy (and exponential family) distribution has two parameterizations: the mean and natural parameterizations $\boldsymbol{\mu}$ and $\boldsymbol{\eta}$. In fact, there exists a bijective mapping called the "forward" mapping $\boldsymbol{\mu} = f(\boldsymbol{\eta})$ (and the "backward" mapping $\boldsymbol{\eta} = f^{-1}(\boldsymbol{\mu})$), which are generally unknown or intractable.

4.5.1.1 Deep inference of maximum entropy distributions

To execute deep inference of a maximum entropy distribution, we must fit a normalizing flow to a distribution satisfying the constraints of Equation 4.62, and select that with maximum entropy. This can be done with a normalizing flow $q_{\boldsymbol{\theta}}$ solving the following equation using the algorithm of maximum entropy flow networks [191]

$$q_{\boldsymbol{\theta}}^*(\mathbf{z}) = \underset{q_{\boldsymbol{\theta}} \in Q}{\text{argmax}}\, H(q_{\boldsymbol{\theta}}(\mathbf{z})) \quad \text{s.t. } \mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\theta}}}\left[T(\mathbf{z})\right] = \boldsymbol{\mu}. \tag{4.64}$$

In deep inference of maximum entropy distributions, the inferred distribution $q_{\boldsymbol{\theta}}^*(\mathbf{z})$ is defined by mean parameter $\boldsymbol{\mu}$. When this statistical inference approach is applied to emergent properties of generative models, the inferred distribution is a posterior in the spirit of variational bayesian inference (see Section 4.5.2.1).

4.5.1.2 Deep inference of exponential families

When the posterior distribution of a bayesian inference problem is known to belong to the exponential family, and its natural parameterization $\boldsymbol{\eta}$ is also known, one can approximate this posterior using variational inference. This is done by optimizing variational parameters $\boldsymbol{\theta}$ of the normalizing flow to approximate the true posterior as closely as possible according to the KL-divergence

$$D(q_{\boldsymbol{\theta}}(\mathbf{z}; \boldsymbol{\eta}) || p(\mathbf{z}; \boldsymbol{\eta})) \tag{4.65}$$

In Chapter 3, we introduced a two network architecture for amortizing variational inference in this family. Over a distribution of the natural parameter $p(\boldsymbol{\eta})$, we optimize the variational fits to the posterior

$$\boldsymbol{\phi}^* = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \, \mathbb{E}_{p(\boldsymbol{\eta})} \left[D(q_{\boldsymbol{\theta}}(\mathbf{z}; \boldsymbol{\eta}) || p(\mathbf{z}; \boldsymbol{\eta})) \right] . \tag{4.66}$$

This algorithm is executed with a two-network architecture, where parameter samples $\mathbf{z}$ are emitted from a normalizing flows with parameters set by a deep function of the natural parameter $\boldsymbol{\theta} = f_{\boldsymbol{\phi}}(\boldsymbol{\eta})$ (Fig. 4.10B). The natural parameter η of the posterior contains hyperparameters of the prior and summary statistics of the data – this is how data enter the EFN architecture.

4.5.2 Variational simulation-based inference

In simulation-based inference, parameter distributions are inferred in complex, simulator-defined models. Historically, approaches to simulation based inference have focused on rejection sampling techniques like approximate bayesian computation (ABC) [51] or markov chain monte carlo (MCMC) [206, 207]. However, in neither of these approaches is a distributional form fit for infer-

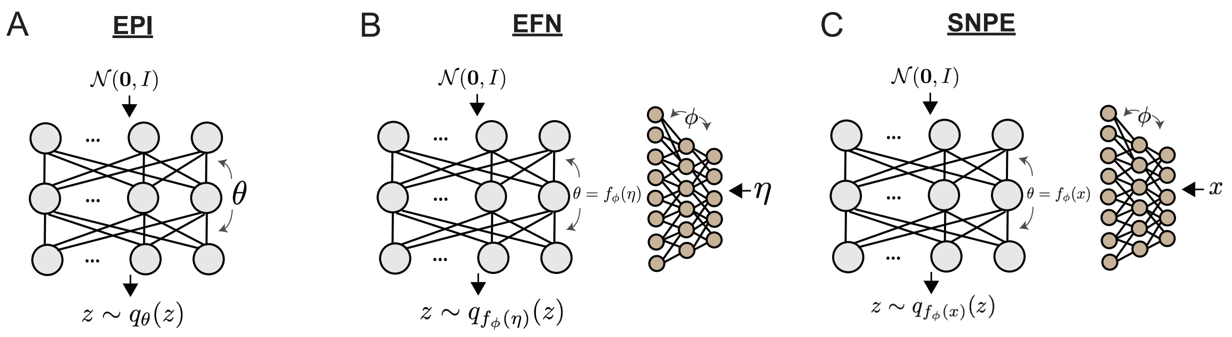

A
EPI
$\mathcal{N}(\mathbf{0}, I)$
θ
$z \sim q_\theta(z)$
B
EFN
$\mathcal{N}(\mathbf{0}, I)$
ϕ
$\theta = f_\phi(\eta)$
η
$z \sim q_{f_\phi(\eta)}(z)$
C
SNPE
$\mathcal{N}(\mathbf{0}, I)$
ϕ
$\theta = f_\phi(x)$
x
$z \sim q_{f_\phi(x)}(z)$

Figure 4.10: Deep inference architectures. **A**. Emergent property inference (EPI) **B**. Exponential family networks (EFN) **C**. Sequential neural posterior estimation (SNPE).

ence. In variational simulation-based inference, a distribution is fit within a parameterized family. Having a variational approximation is important in the context of scientific research, where we want to efficiently quantify the structure of the inferred distribution.

4.5.2.1 Emergent property inference

In emergent property inference (EPI), a maximum entropy distribution of model parameters is inferred such that they produce an emergent property (see Chapter 4.). An emergent property is quantified as statistical constraints on data that emerge from the generative model of interest. EPI is fit using the MEFN algorithm, where $T(\mathbf{z}) = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}\sim\mathbf{z}} \left[\phi(\mathbf{z})\right]$ (see Algorithm 1). In EPI, only a single deep neural network in the normalizing flow is used in the learning architecture (Fig. 4.10A).

4.5.2.2 Sequential neural posterior estimation

In sequential neural posterior estimation (SNPE), an inference network $f_{\boldsymbol{\phi}}(\mathbf{x})$ learns to map data to variational parameters of the approximate posterior $q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})$. This is done in the first optimization epoch by optimizing a density estimation objective

$$\boldsymbol{\phi}^* = \underset{\boldsymbol{\phi}}{\operatorname{argmax}}\, \mathbb{E}_{\mathbf{z},\mathbf{x}} \left[\log q_{f_{\boldsymbol{\phi}}(\mathbf{x})}(\mathbf{z})\right], \tag{4.67}$$

where paired samples $(\mathbf{x}_i, \mathbf{z}_i)$ are sampled from the joint distribution $p(\mathbf{x}, \mathbf{z})$. In subsequent epochs (*sequential* epochs), paired samples are produced by sampling from the current approximate posterior $\mathbf{z}_i \sim q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})$ and from the model simulator $\mathbf{x}_i \sim p(\mathbf{x} \mid \mathbf{z})$. The two-network architecture for SNPE is shown in Figure 4.10C.

4.5.2.3 Sequential optimization

Both EPI and SNPE update their optimization objectives in sequential epochs based on their latest approximations, however the nature of these updates are very different. In EPI, the optimal lagrangian penalty coefficients $\boldsymbol{\eta}_{\text{opt}}$ are estimated from the current constraint errors. The lagrangian penalty coefficients are adjusted in each epoch of the EPI optimization until convergence, where all constraints of the emergent property are satisfied.

In SNPE, the distribution of paired samples $(\mathbf{x}_i, \mathbf{z}_i)$ over which the density estimation objective is optimized is adjusted to be closer to the posterior with each epoch. At the beginning of each SNPE epoch, the current estimate of the posterior is sampled to obtain many new parameters $\mathbf{z}_i$ and paired simulations $\mathbf{x}_i$ to augment the current training dataset. In both the EPI and SNPE algorithms, a period of stochastic gradient descent on a consistent objective is followed by an adaptation of the objective function of the optimization.

4.5.3 Two-network architectures for deep inference

While EPI has a single network in its architecture, SNPE has two networks in the same manner as EFNs. This gives SNPE the capacity for amortization, a relative advantage over EPI. If there was a simple bijective mapping from the mean parameter $\boldsymbol{\mu}$ of EPI to the natural parameter $\boldsymbol{\eta}$, we could use the approach of EFN to amortize inference of emergent properties. However, as we discussed in Section 4.5.2.1, the backward mapping is formally hard for general maximum entropy distributions.

Exponential family posteriors have consistent form and dimensionality of natural parameter $\boldsymbol{\eta}$ no matter the number of observed data points. Once trained, an EFN can return a posterior approximation for datasets of varying observation count (Equation 3.3). In SNPE, this is not the case. The data conditioned upon $\mathbf{x}_i$ is either a concatenation of a static number of observations, or a summary statistic calculated over a static number of observations.

Chapter 5

Chapter 5: Building theories of neural circuits with emergent property inference

In Chapter 4, we introduced a deep generative modeling technique for inferring distributions of neural circuit model parameters that produce emergent properties. This chapter corresponds to selected sections of Bittner et al. 2021 [163] related to the application of EPI to models of primary visual cortex and superior colliculus. By fitting normalizing flows to parameter distributions of these models that produce emergent properties of computation, we gained novel insight, which builds upon mechanistic theories of these neural circuits. The remainder of this chapter was co-authored by Agostina Palmigiano, Alex T. Piet, Chunyu A. Duan, Carlos D. Brody, Kenneth D. Miller, and John P. Cunningham.

5.1 EPI reveals how recurrence with multiple inhibitory subtypes governs excitatory variability in a V1 model

Dynamical models of excitatory (E) and inhibitory (I) populations with supralinear input-output function have succeeded in explaining a host of experimentally documented phenomena in primary visual cortex (V1). In a regime characterized by inhibitory stabilization of strong recurrent excitation, these models give rise to paradoxical responses [170], selective amplification [200, 194], surround suppression [226] and normalization [227]. Recent theoretical work [228] shows that stabilized E-I models reproduce the effect of variability suppression [229]. Furthermore, experimental evidence shows that inhibition is composed of distinct elements – parvalbumin (P), somatostatin (S), VIP (V) – composing 80% of GABAergic interneurons in V1 [230, 231, 232], and that these inhibitory cell types follow specific connectivity patterns (Fig. 5.1A) [233]. Here, we use EPI on a model of V1 with biologically realistic connectivity to show how the structure of input across neuron types affects the variability of the excitatory population – the population largely responsible for projecting to other brain areas [234].

We considered response variability of a nonlinear dynamical V1 circuit model (Fig. 5.1A) with a state comprised of each neuron-type population's rate $\mathbf{x} = [x_E, x_P, x_S, x_V]^\top$. Each population receives recurrent input $W\mathbf{x}$, where W is the effective connectivity matrix (see Section 5.5.1) and an external input with mean $\mathbf{h}$, which determines population rate via supralinear nonlinearity $\phi(\cdot) = [\cdot]_+^2$. The external input has an additive noisy component $\boldsymbol{\epsilon}$ with variance $\boldsymbol{\sigma}^2 = \left[\sigma_E^2, \sigma_P^2, \sigma_S^2, \sigma_V^2\right]$. This noise has a slower dynamical timescale $\tau_{\text{noise}} > \tau$ than the population rate, allowing fluctuations around a stimulus-dependent steady-state (Fig. 5.1B). This model is the stochastic stabilized supralinear network (SSSN) [228]

$$\tau \frac{d\mathbf{x}}{dt} = -\mathbf{x} + \phi(W\mathbf{x} + \mathbf{h} + \boldsymbol{\epsilon}), \tag{5.1}$$

generalized to have multiple inhibitory neuron types. It introduces stochasticity to four neuron-type models of V1 [197]. Stochasticity and inhibitory multiplicity introduce substantial complexity to the mathematical treatment of this problem (see Section 5.5.1.5) motivating the analysis of this model with EPI. Here, we consider fixed weights W and input $\mathbf{h}$ [198], and study the effect of input variability $\mathbf{z} = [\sigma_E, \sigma_P, \sigma_S, \sigma_V]^\top$ on excitatory variability.

We quantify levels of E-population variability by studying two emergent properties

$$\begin{aligned} \mathcal{X}(5\text{Hz}) \ : \ \mathbb{E}_{\mathbf{z},\mathbf{x}}\left[s_E(\mathbf{x};\mathbf{z})\right] &= 5\text{Hz} \qquad \mathcal{X}(10\text{Hz}) \ : \ \mathbb{E}_{\mathbf{z},\mathbf{x}}\left[s_E(\mathbf{x};\mathbf{z})\right] = 10\text{Hz} \\ \text{Var}_{\mathbf{z},\mathbf{x}}\left[s_E(\mathbf{x};\mathbf{z})\right] &= 1\text{Hz}^2 \qquad \text{Var}_{\mathbf{z},\mathbf{x}}\left[s_E(\mathbf{x};\mathbf{z})\right] = 1\text{Hz}^2, \end{aligned} \tag{5.2}$$

where $s_E(\mathbf{x};\mathbf{z})$ is the standard deviation of the stochastic E-population response about its steady state (Fig. 5.1C). In the following analyses, we select 1Hz^2 variance such that the two emergent properties do not overlap in $s_E(\mathbf{z};\mathbf{x})$.

First, we ran EPI to obtain parameter distribution $q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathcal{X}(5\text{Hz}))$ producing E-population variability around 5Hz (Fig. 5.1D). From the marginal distribution of σ_E and σ_P (Fig. 5.1D, top-

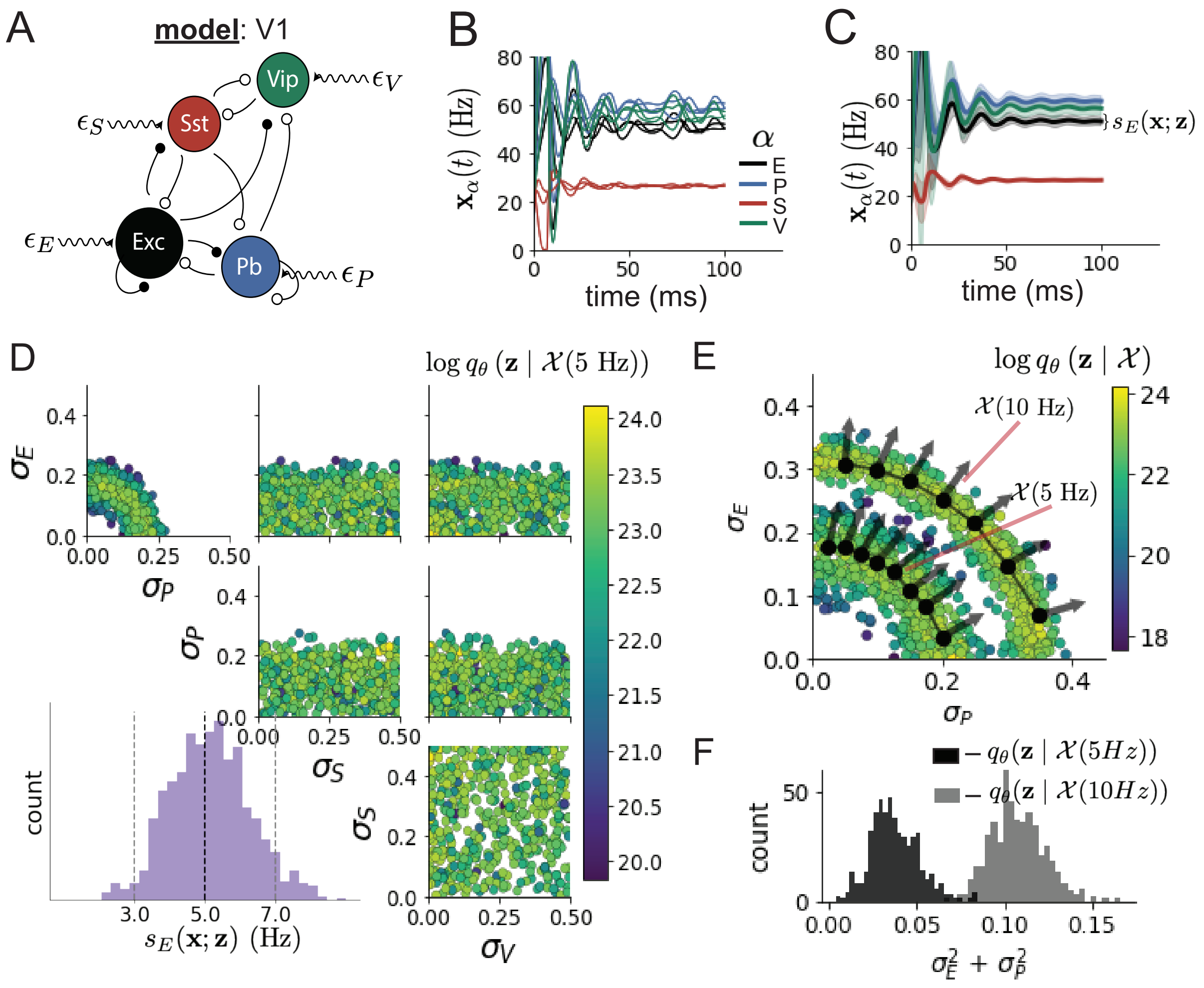


Figure 5.1: Emergent property inference in the stochastic stabilized supralinear network (SSSN) **A**. Four-population model of primary visual cortex with excitatory (black), parvalbumin (blue), somatostatin (red), and VIP (green) neurons (excitatory and inhibitory projections filled and unfilled, respectively). Some neuron-types largely do not form synaptic projections to others ($|W_{\alpha_1,\alpha_2})| < 0.025$). Each neural population receives a baseline input $\mathbf{h}_b$, and the E- and P-populations also receive a contrast-dependent input $\mathbf{h}_c$. Additionally, each neural population receives a slow noisy input $\boldsymbol{\epsilon}$. **B**. Transient network responses of the SSSN model. Traces are independent trials with varying initialization $\mathbf{x}(0)$ and noise $\boldsymbol{\epsilon}$. **C**. Mean (solid line) and standard deviation $s_E(\mathbf{x};\mathbf{z})$ (shading) across 100 trials. **D**. EPI distribution of noise parameters $\mathbf{z}$ conditioned on E-population variability. The EPI predictive distribution of $s_E(\mathbf{x};\mathbf{z})$ is show on the bottom-left. **E**. (Top) Enlarged visualization of the σ_E-σ_P marginal distribution of EPI $q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathcal{X}(5\text{Hz}))$ and $q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathcal{X}(10\text{Hz}))$. Each black dot shows the mode at each σ_P. The arrows show the most sensitive dimensions of the Hessian evaluated at these modes. **F**. The predictive distributions of $\sigma_E^2 + \sigma_P^2$ of each inferred distribution $q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathcal{X}(5\text{Hz}))$ and $q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathcal{X}(10\text{Hz}))$.

left), we can see that $s_E(\mathbf{x}; \mathbf{z})$ is sensitive to various combinations of σ_E and σ_P. Alternatively, both σ_S and σ_V are degenerate with respect to $s_E(\mathbf{x}; \mathbf{z})$ evidenced by the unexpectedly high variability in those dimensions (Fig. 5.1D, bottom-right). Together, these observations imply a curved path with respect to $s_E(\mathbf{x}; \mathbf{z})$ of 5Hz, which is indicated by the modes along σ_P (Fig. 5.1E).

Figure 5.1E suggests a quadratic relationship in E-population fluctuations and the standard deviation of E- and P-population input; as the square of either σ_E or σ_P increases, the other compensates by decreasing to preserve the level of $s_E(\mathbf{x}; \mathbf{z})$. This quadratic relationship is preserved at greater level of E-population variability $\mathcal{X}(10\text{Hz})$ (Fig. 5.1E and 5.4). Indeed, the sum of squares of σ_E and σ_P is larger in $q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathcal{X}(10\text{Hz}))$ than $q_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathcal{X}(5\text{Hz}))$ (Fig 5.1F, $p < 1 \times 10^{-10}$), while the sum of squares of σ_S and σ_V are not significantly different in the two EPI distributions (Fig. 5.6, $p = .40$), in which parameters were bounded from 0 to 0.5. The strong interaction between E- and P-population input variability on excitatory variability is intriguing, since this circuit exhibits a paradoxical effect in the P-population (and no other inhibitory types) (Fig. 5.7), meaning that the E-population is P-stabilized. Future research may uncover a link between the population of network stabilization and compensatory interactions governing excitatory variability.

EPI revealed the quadratic dependence of excitatory variability on input variability to the E- and P-populations, as well as its independence to input from the other two inhibitory populations. In a simplified model ($\tau = \tau_{\text{noise}}$), it can be shown that surfaces of equal variance are ellipsoids as a function of $\boldsymbol{\sigma}$ (see Section 5.5.1.5). Nevertheless, the sensitive and degenerate parameters are intractable to predict mathematically, since the covariance matrix depends on the steady-state solution of the network [228, 235], and terms in the covariance expression increase quadratically with each additional neuron-type population (see also Section 5.5.1.5). By pointing out this mathematical complexity, we emphasize the value of EPI for gaining understanding about theoretical models when mathematical analysis becomes onerous or impractical.

5.2 EPI identifies two regimes of rapid task switching

It has been shown that rats can learn to switch from one behavioral task to the next on randomly interleaved trials [236], and an important question is what neural mechanisms produce this computation. In this experimental setup, rats were given an explicit task cue on each trial, either Pro or Anti. After a delay period, rats were shown a stimulus, and made a context (task) dependent response (Fig. 5.2A). In the Pro task, rats were required to orient towards the stimulus, while in the Anti task, rats were required to orient away from the stimulus. Pharmacological inactivation of the SC impaired rat performance, and time-specific optogenetic inactivation revealed a crucial role for the SC on the cognitively demanding Anti trials [199]. These results motivated a nonlinear dynamical model of the SC containing four functionally-defined neuron-type populations. In Duan et al. 2021, a computationally intensive procedure was used to obtain a set of 373 connectivity parameters that qualitatively reproduced these optogenetic inactivation results. To build upon the insights of this previous work, we use the probabilistic tools afforded by EPI to identify and characterize two linked, yet distinct regimes of rapid task switching connectivity.

In this SC model, there are Pro- and Anti-populations in each hemisphere (left (L) and right (R)) with activity variables $\mathbf{x} = [x_{LP}, x_{LA}, x_{RP}, x_{RA}]^\top$ [199]. The connectivity of these populations is parameterized by self sW, vertical vW, diagonal dW and horizontal hW connections (Fig. 5.2B). The input $\mathbf{h}$ is comprised of a positive cue-dependent signal to the Pro or Anti populations, a positive stimulus-dependent input to either the Left or Right populations, and a choice-period input to the entire network (see Section 5.5.2.1). Model responses are bounded from 0 to 1 as a function ϕ of an internal variable $\mathbf{u}$

$$\begin{aligned} \tau \frac{d\mathbf{u}}{dt} &= -\mathbf{u} + W\mathbf{x} + \mathbf{h} + d\mathbf{B} \\ \mathbf{x} &= \phi(\mathbf{u}). \end{aligned} \tag{5.3}$$

The model responds to the side with greater Pro neuron activation; e.g. the response is left if $x_{LP} > x_{RP}$ at the end of the trial. Here, we use EPI to determine the network connectivity $\mathbf{z} =$

$[sW, vW, dW, hW]^\top$ that produces rapid task switching.

Rapid task switching is formalized mathematically as an emergent property with two statistics: accuracy in the Pro task $p_P(\mathbf{x}; \mathbf{z})$ and Anti task $p_A(\mathbf{x}; \mathbf{z})$. We stipulate that accuracy be on average .75 in each task with variance $.075^2$

$$\begin{aligned} \mathcal{X} \; : \; \mathbb{E}_{\mathbf{z}} \begin{bmatrix} p_P(\mathbf{x}; \mathbf{z}) \\ p_A(\mathbf{x}; \mathbf{z}) \end{bmatrix} &= \begin{bmatrix} .75 \\ .75 \end{bmatrix} \\ \mathrm{Var}_{\mathbf{z}} \begin{bmatrix} p_P(\mathbf{x}; \mathbf{z}) \\ p_A(\mathbf{x}; \mathbf{z}) \end{bmatrix} &= \begin{bmatrix} .075^2 \\ .075^2 \end{bmatrix}. \end{aligned} \tag{5.4}$$

75% accuracy is a realistic level of performance in each task, and with the chosen variance, inferred models will not exhibit fully random responses (50%), nor perfect performance (100%).

The EPI inferred distribution (Fig. 5.2C) produces Pro and Anti task accuracies (Fig. 5.2C, bottom-left) consistent with rapid task switching (Equation 5.4). This parameter distribution has rich structure that is not captured well by simple linear correlations (Fig. 5.8). Specifically, the shape of the EPI distribution is sharply bent, matching ground truth structure indicated by brute-force sampling (Fig. 5.14). This is most saliently observed in the marginal distribution of sW-hW (Fig. 5.2C top-right), where anticorrelation between sW and hW switches to correlation with decreasing sW. By identifying the modes of the EPI distribution $\mathbf{z}^*(sW)$ at different values of sW (Fig. 5.2C red/purple dots), we can quantify this change in distributional structure with the sensitivity dimension $\mathbf{v}_1(\mathbf{z})$ (Fig. 5.2C red/purple arrows). Note that the directionality of these sensitivity dimensions at $\mathbf{z}^*(sW)$ changes distinctly with sW, and are perpendicular to the robust dimensions of the EPI distribution that preserve rapid task switching. These two directionalities of sensitivity motivate the distinction of connectivity into two regimes, which produce different types of responses in the Pro and Anti tasks (Fig. 5.9).

When perturbing connectivity along the sensitivity dimension away from the modes

$$\mathbf{z} = \mathbf{z}^*(sW) + \delta \mathbf{v}_1(\mathbf{z}^*(sW)), \tag{5.5}$$

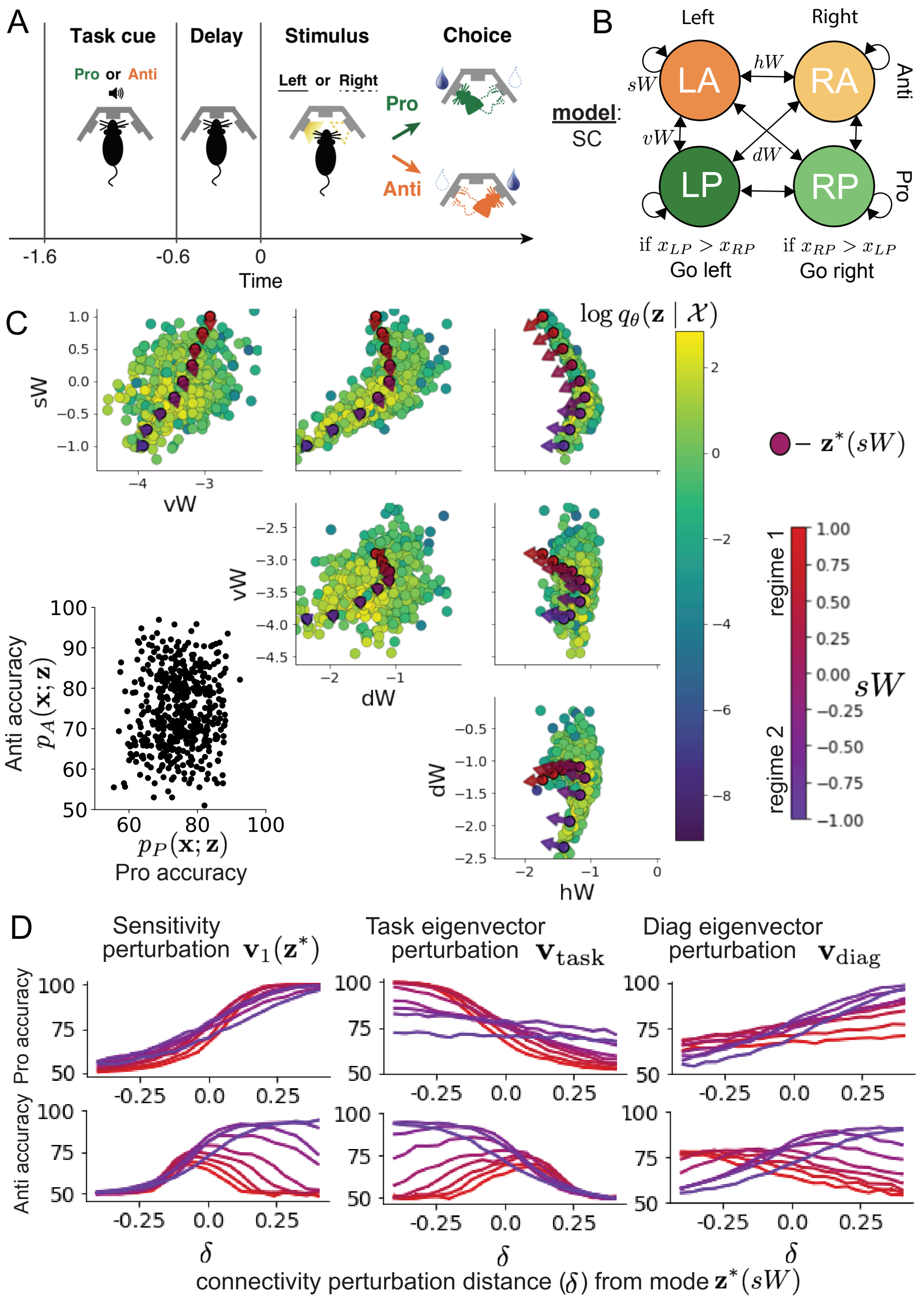


Figure 5.2: **A**. Rapid task switching behavioral paradigm (see text). **B**. Model of superior colliculus (SC). Neurons: LP - Left Pro, RP - Right Pro, LA - Left Anti, RA - Right Anti. Parameters: sW - self, hW - horizontal, vW -vertical, dW - diagonal weights. **C**. The EPI inferred distribution of rapid task switching networks. Red/purple parameters indicate modes $\mathbf{z}^*(sW)$ colored by sW. Sensitivity vectors $\mathbf{v}_1(\mathbf{z}^*)$ are shown by arrows. (Bottom-left) EPI predictive distribution of task accuracies. **D**. Mean and standard error ($N_{\text{test}} = 25$, bars not visible) of accuracy in Pro (top) and Anti (bottom) tasks after perturbing connectivity away from mode along $\mathbf{v}_1(\mathbf{z}^*)$ (left), $\mathbf{v}_{\text{task}}$ (middle), and $\mathbf{v}_{\text{diag}}$ (right).

Pro accuracy monotonically increases in both regimes (Fig. 5.2D, top-left). However, there is a stark difference between regimes in Anti accuracy. Anti accuracy falls in either direction of $\mathbf{v}_1$ in regime 1, yet monotonically increases along with Pro accuracy in regime 2 (Fig. 5.2D, bottom-left). The sharp change in local structure of the EPI distribution is therefore explained by distinct sensitivities: Anti accuracy diminishes in only one or both directions of the sensitivity perturbation.

To understand the mechanisms differentiating the two regimes, we can make connectivity perturbations along dimensions that only modify a single eigenvalue of the connectivity matrix. These eigenvalues λ_{all}, λ_{side}, λ_{task}, and λ_{diag} correspond to connectivity eigenmodes with intuitive roles in processing in this task (Fig. 5.10A). For example, greater λ_{task} will strengthen internal representations of task, while greater λ_{diag} will amplify dominance of Pro and Anti pairs in opposite hemispheres (Section 5.5.2.7). Unlike the sensitivity dimension, the dimensions $\mathbf{v}_a$ that perturb isolated connectivity eigenvalues λ_a for $a \in \{\text{all}, \text{side}, \text{task}, \text{diag}\}$ are independent of $\mathbf{z}^*(sW)$ (see Section 5.5.2.7), e.g.

$$\mathbf{z} = \mathbf{z}^*(sW) + \delta \mathbf{v}_{\text{task}}. \tag{5.6}$$

Connectivity perturbation analyses reveal that decreasing λ_{task} has a very similar effect on Anti accuracy as perturbations along the sensitivity dimension (Fig. 5.2D, middle). The similar effects of perturbations along the sensitivity dimension $\mathbf{v}_1(\mathbf{z}^*)$ and reduction of task eigenvalue (via perturbations along $-\mathbf{v}_{\text{task}}$) suggest that there is a carefully tuned strength of task representation in connectivity regime 1, which if disturbed results in random Anti trial responses. Finally, we recognize that increasing λ_{diag} has opposite effects on Anti accuracy in each regime (Fig. 5.2D, right). In the next section, we build on these mechanistic characterizations of each regime by examining their resilience to optogenetic inactivation.

5.3 EPI inferred SC connectivities reproduce results from optogenetic inactivation experiments

During the delay period of this task, the circuit must prepare to execute the correct task according to the presented cue. The circuit must then maintain a representation of task throughout the delay period, which is important for correct execution of the Anti task. Duan et al. found that bilateral optogenetic inactivation of SC during the delay period consistently decreased performance in the Anti task, but had no effect on the Pro task (Fig. 5.3A) [199]. The distribution of connectivities inferred by EPI exhibited this same effect in simulation at high optogenetic strengths γ, which reduce the network activities $\mathbf{x}(t)$ by a factor $1 - \gamma$ (Fig. 5.3B) (see Section 5.5.2.8).

To examine how connectivity affects response to delay period inactivation, we grouped connectivities of the EPI distribution along the continuum linking regimes 1 and 2 of Section 5.2. $Z(sW)$ is the set of EPI samples for which the closest mode was $\mathbf{z}^*(sW)$ (see Section 5.5.2.4). In the following analyses, we examine how error, and the influence of connectivity eigenvalue on Anti error change along this continuum of connectivities. Obtaining the parameter samples for these analysis with the learned EPI distribution was more than 20,000 times faster than a brute force approach (see Section 5.5.2.5).

The mean increase in Anti error of the EPI distribution is closest to the experimentally measured value of 7% at $\gamma = 0.675$ (Fig. 5.3B, black dot). At this level of optogenetic strength, regime 1 exhibits an increase in Anti error with delay period silencing (Fig. 5.3C, left), while regime 2 does not. In regime 1, greater λ_{task} and λ_{diag} decrease Anti error (Fig. 5.3C, right). In other words, stronger task representations and diagonal amplification make the SC model more resilient to delay period silencing in the Anti task. This complements the finding from Duan et al. 2021 [199] that λ_{task} and λ_{diag} improve Anti accuracy.

At roughly $\gamma = 0.85$ (Fig. 5.3B, gray dot), the Anti error saturates, while Pro error remains at zero. Following delay period inactivation at this optogenetic strength, there are strong similarities in the responses of Pro and Anti trials during the choice period (Fig. 5.3D, left). We interpreted

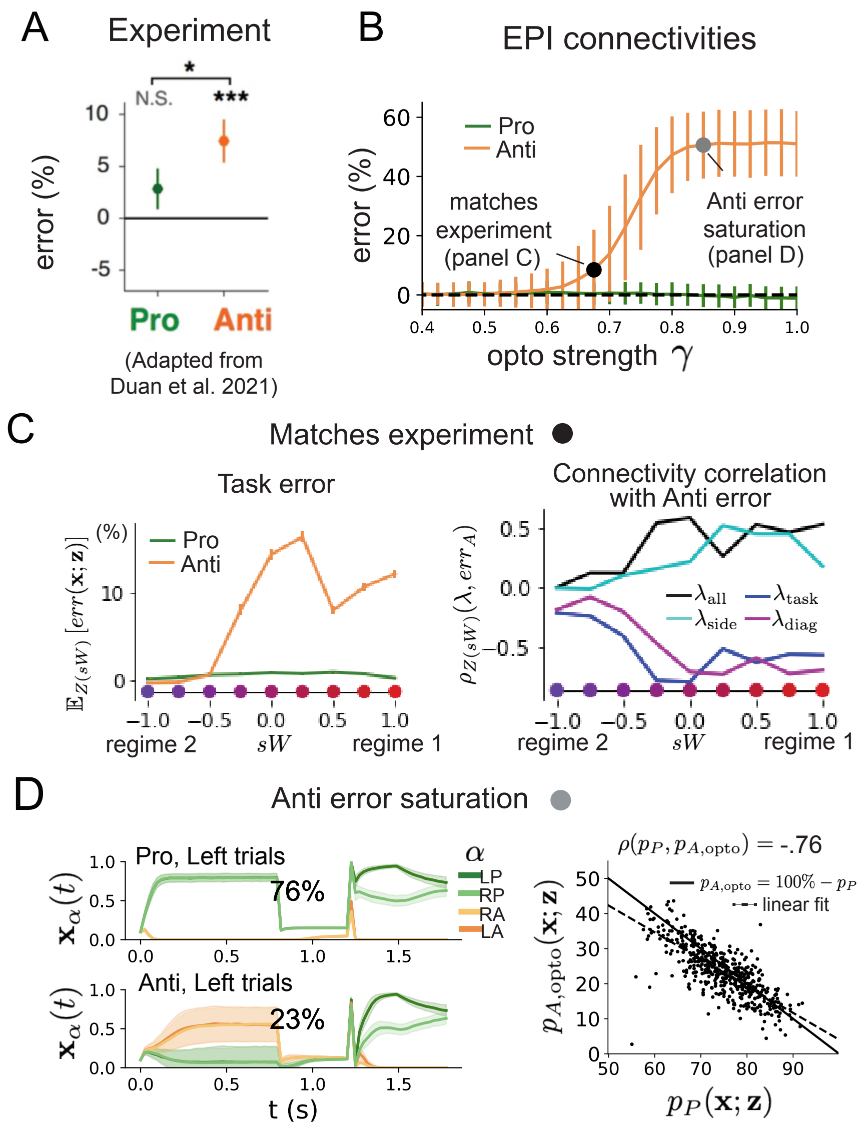


Figure 5.3: **A**. Mean and standard error (bars) across recording sessions of task error following delay period optogenetic inactivation in rats. **B**. Mean and standard deviation (bars) of task error induced by delay period inactivation of varying optogenetic strength γ across the EPI distribution. **C**. (Left) Mean and standard error of Pro and Anti error from regime 1 to regime 2 at $\gamma = 0.675$. (Right) Correlations of connectivity eigenvalues with Anti error from regime 1 to regime 2 at $\gamma = 0.675$. **D**. (Left) Mean and standard deviation (shading) of responses of the SC model at the mode of the EPI distribution to delay period inactivation at $\gamma = 0.85$. Accuracy in Pro (top) and Anti (bottom) task is shown as a percentage. (Right) Anti accuracy following delay period inactivation at $\gamma = 0.85$ versus accuracy in the Pro task across connectivities in the EPI distribution.

these similarities to suggest that delay period inactivation at this saturated level flips the internal representation of task (from Anti to Pro) in the circuit model. A flipped task representation would explain why the Anti error saturates at 50%: the average Anti accuracy in EPI inferred connectivities is 75%, but average Anti accuracy would be 25% (100% - $\mathbb{E}_{\mathbf{z}}\left[p_P\right]$) if the internal representation of task is flipped during the delay period. This hypothesis prescribes a model of Anti accuracy during delay period silencing of $p_{A,\text{opto}} = 100\% - p_P$, which is fit closely across both regimes of the EPI inferred connectivities (Fig. 5.3D, right). Similarities between Pro and Anti trial responses were not present at the experiment-matching level of $\gamma = 0.675$ (Fig. 5.12 left) and neither was anticorrelation in p_P and $p_{A,\text{opto}}$ (Fig. 5.12 right).

In summary, the connectivity inferred by EPI to perform rapid task switching replicated results from optogenetic silencing experiments. We found that at levels of optogenetic strength matching experimental levels of Anti error, only one regime actually exhibited the effect. This connectivity regime is less resilient to optogenetic perturbation, and perhaps more biologically realistic. Finally, we characterized the pathology in Anti error that occurs in both regimes when optogenetic strength is increased to high levels, leading to a mechanistic hypothesis that is experimentally testable. The probabilistic tools afforded by EPI yielded this insight: we identified two regimes and the continuum of connectivities between them by taking gradients of parameter probabilities in the EPI distribution, we identified sensitivity dimensions by measuring the Hessian of the EPI distribution, and we obtained many parameter samples at each step along the continuum at an efficient rate.

5.4 Discussion

In this paper, we demonstrate the value of deep inference for parameter sensitivity analyses at both the local and global level. With these techniques, flexible deep probability distributions are optimized to capture global structure by approximating the full distribution of suitable parameters. Importantly, the local structure of this deep probability distribution can be quantified at any parameter choice, offering instant sensitivity measurements after fitting. For example, the global structure captured by EPI revealed two distinct parameter regimes, which had different local structure quantified by the deep probability distribution (see Section 5.5.2). In comparison, bayesian MCMC is considered a popular approach for capturing global parameter structure [208], but there is no variational approximation (the deep probability distribution in EPI), so sensitivity information is not queryable and sampling remains slow after convergence. Local sensitivity analyses (e.g. [182]) may be performed independently at individual parameter samples, but these methods alone do not capture the full picture in nonlinear, complex distributions. In contrast, deep inference yields a probability distribution that produces a wholistic assessment of parameter sensitivity at the local and global level, which we used in this study to make novel insights into a range of theoretical models. Together, the abilities to condition upon emergent properties, the efficient inference algorithm, and the capacity for parameter sensitivity analyses make EPI a useful method for addressing inverse problems in theoretical neuroscience.

Acknowledgements:
This work was funded by NSF Graduate Research Fellowship, DGE-1644869, McKnight Endowment Fund, NIH NINDS 5R01NS100066, Simons Foundation 542963, NSF NeuroNex Award, DBI-1707398, The Gatsby Charitable Foundation, Simons Collaboration on the Global Brain Postdoctoral Fellowship, Chinese Postdoctoral Science Foundation, and International Exchange Program Fellowship. We also acknowledge the Marine Biological Laboratory Methods in Computational Neuroscience Course, where this work was discussed and explored in its early stages. Helpful conversations were had with Larry Abbott, Stephen Baccus, James Fitzgerald, Gabrielle Gutierrez, Francesca Mastrogiuseppe, Srdjan Ostojic, Liam Paninski, and Dhruva Raman.

Data availability statement:
The datasets generated during this study have been made publicly available on Zenodo at this address: https://doi.org/10.5281/zenodo.4910010 .

Code availability statement:
All software written for the current study is available at https://github.com/cunningham-lab/epi.

5.5 Methods

5.5.1 Primary visual cortex

5.5.1.1 V1 model

E-I circuit models, rely on the assumption that inhibition can be studied as an indivisible unit, despite ample experimental evidence showing that inhibition is instead composed of distinct elements [232]. In particular three types of genetically identified inhibitory cell-types – parvalbumin (P), somatostatin (S), VIP (V) – compose 80% of GABAergic interneurons in V1 [230, 231, 232], and follow specific connectivity patterns (Fig. 5.1A) [233], which lead to cell-type specific computations [237, 198]. Currently, how the subdivision of inhibitory cell-types, shapes correlated variability by reconfiguring recurrent network dynamics is not understood.

In the stochastic stabilized supralinear network [228], population rate responses $\mathbf{x}$ to mean input $\mathbf{h}$, recurrent input $W\mathbf{x}$ and slow noise $\boldsymbol{\epsilon}$ are governed by

$$\tau \frac{d\mathbf{x}}{dt} = -\mathbf{x} + \phi(W\mathbf{x} + \mathbf{h} + \boldsymbol{\epsilon}), \tag{5.7}$$

where the noise is an Ornstein-Uhlenbeck process $\boldsymbol{\epsilon} \sim OU(\tau_{\text{noise}}, \boldsymbol{\sigma})$

$$\tau_{\text{noise}} d\epsilon_\alpha = -\epsilon_\alpha dt + \sqrt{2\tau_{\text{noise}}}\tilde{\sigma}_\alpha dB \tag{5.8}$$

with $\tau_{\text{noise}} = 5\text{ms} > \tau = 1\text{ms}$. The noisy process is parameterized as

$$\tilde{\sigma}_\alpha = \sigma_\alpha \sqrt{1 + \frac{\tau}{\tau_{\text{noise}}}}, \tag{5.9}$$

so that $\boldsymbol{\sigma}$ parameterizes the variance of the noisy input in the absence of recurrent connectivity ($W = \mathbf{0}$). As contrast $c \in [0, 1]$ increases, input to the E- and P-populations increases relative to a baseline input $\mathbf{h} = \mathbf{h}_b + c\mathbf{h}_c$. Connectivity ($W_{\text{fit}}$) and input ($\mathbf{h}_{b,\text{fit}}$ and $\mathbf{h}_{c,\text{fit}}$) parameters were fit using the deterministic V1 circuit model [198]

$$W_{\text{fit}} = \begin{bmatrix} W_{EE} & W_{EP} & W_{ES} & W_{EV} \\ W_{PE} & W_{PP} & W_{PS} & W_{PV} \\ W_{SE} & W_{SP} & W_{SS} & W_{SV} \\ W_{VE} & W_{VP} & W_{VS} & W_{VV} \end{bmatrix} = \begin{bmatrix} 2.18 & -1.19 & -.594 & -.229 \\ 1.66 & -.651 & -.680 & -.242 \\ .895 & -5.22\times10^{-3} & -1.51\times10^{-4} & -.761 \\ 3.34 & -2.31 & -.254 & -2.52\times10^{-4} \end{bmatrix}, \tag{5.10}$$

$$\mathbf{h}_{b,\text{fit}} = \begin{bmatrix} .416 \\ .429 \\ .491 \\ .486 \end{bmatrix}, \tag{5.11}$$

and

$$\mathbf{h}_{c,\text{fit}} = \begin{bmatrix} .359 \\ .403 \\ 0 \\ 0 \end{bmatrix}. \tag{5.12}$$

To obtain rates on a realistic scale (100-fold greater), we map these fitted parameters to an equivalence class

$$W = \begin{bmatrix} W_{EE} & W_{EP} & W_{ES} & W_{EV} \\ W_{PE} & W_{PP} & W_{PS} & W_{PV} \\ W_{SE} & W_{SP} & W_{SS} & W_{SV} \\ W_{VE} & W_{VP} & W_{VS} & W_{VV} \end{bmatrix} = \begin{bmatrix} .218 & -.119 & -.0594 & -.0229 \\ .166 & -.0651 & -.068 & -.0242 \\ .0895 & -5.22\times10^{-4} & -1.51\times10^{-5} & -.0761 \\ .334 & -.231 & -.0254 & -2.52\times10^{-5} \end{bmatrix}, \tag{5.13}$$

$$\mathbf{h}_b = \begin{bmatrix} h_{b,E} \\ h_{b,P} \\ h_{b,S} \\ h_{b,V} \end{bmatrix} = \begin{bmatrix} 4.16 \\ 4.29 \\ 4.91 \\ 4.86 \end{bmatrix}, \tag{5.14}$$

and

$$\mathbf{h}_c = \begin{bmatrix} h_{c,E} \\ h_{c,P} \\ h_{c,S} \\ h_{c,V} \end{bmatrix} = \begin{bmatrix} 3.59 \\ 4.03 \\ 0 \\ 0 \end{bmatrix}. \tag{5.15}$$

Circuit responses are simulated using $T = 200$ time steps at $dt = 0.5\text{ms}$ from an initial condition drawn from $\mathbf{x}(0) \sim U[10\text{Hz}, 25\text{Hz}]$. Standard deviation of the E-population $s_E(\mathbf{x}; \mathbf{z})$ is calculated as the square root of the temporal variance from $t_{ss} = 75\text{ms}$ to $Tdt = 100\text{ms}$

$$s_E(\mathbf{x}; \mathbf{z}) = \sqrt{\mathbb{E}_{t>t_{ss}}\left[\left(x_E(t) - \mathbb{E}_{t>t_{ss}}\left[x_E(t)\right]\right)^2\right]}. \tag{5.16}$$

5.5.1.2 EPI details for the V1 model

To write the emergent properties of Equation 5.2 in terms of the EPI optimization, we have

$$f(\mathbf{x}; \mathbf{z}) = s_E(\mathbf{x}; \mathbf{z}), \tag{5.17}$$

$$\boldsymbol{\mu} = \begin{bmatrix} 5 \end{bmatrix} \tag{5.18}$$

(or $\boldsymbol{\mu} = \begin{bmatrix} 10 \end{bmatrix}$), and

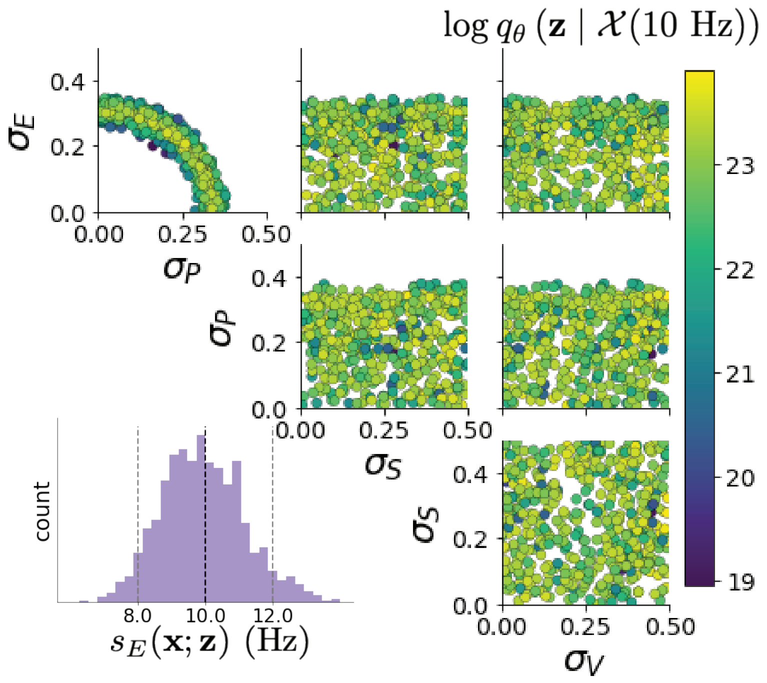
$$\sigma^2 = \begin{bmatrix} 1^2 \end{bmatrix} \tag{5.19}$$

(see Sections 4.4.1.3-4.4.1.4, and example in Section 4.4.1.5).

For EPI in Figures 5.1D-E and 5.4, we used a real NVP architecture with three coupling layers and two-layer neural networks of 50 units per layer. The normalizing flow architecture mapped

$z_0 \sim \mathcal{N}(\mathbf{0}, I)$ to a support of $\mathbf{z} = [\sigma_E, \sigma_P, \sigma_S, \sigma_V] \in [0.0, 0.5]^4$. EPI optimization was run using three different random seeds for architecture initialization $\boldsymbol{\theta}$ with an augmented lagrangian coefficient of $c_0 = 10^{-1}$, $\beta = 2$, a batch size $n = 100$, and simulated 100 trials to calculate average $s_E(\mathbf{x}; \mathbf{z})$ for each $\mathbf{z}^{(i)}$. We used $i_{\text{max}} = 2,000$ iterations per epoch. The distributions shown are those of the architectures converging with criteria $N_{\text{test}} = 100$ at greatest entropy across three random seeds. Optimization details are shown in Figure 5.5. The sums of squares of each pair of parameters are shown for each EPI distribution in Figure 5.6. The plots are histograms of 500 samples from each EPI distribution from which the significance p-values of Section 5.1 are determined.

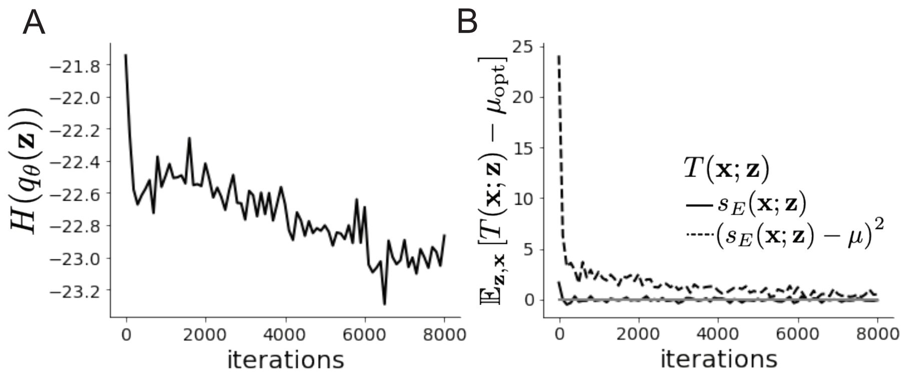
Figure 5.4: EPI inferred distribution for $\mathcal{X}(10\text{Hz})$.



5.5.1.3 Sensitivity analyses

In Fig. 5.1E, we visualize the modes of $q_\theta(\mathbf{z} \mid \mathcal{X})$ throughout the σ_E-σ_P marginal. At each local mode $\mathbf{z}^*(\sigma_P)$, where σ_P is fixed, we calculated the Hessian and visualized the sensitivity dimension in the direction of positive σ_E.

Figure 5.5: EPI optimization $q_\theta(\mathbf{z} \mid \mathcal{X}(5\text{Hz}))$ **A**. Entropy throughout optimization. **B**. The emergent property statistic means and variances converge to their constraints at 8,000 iterations following the fourth augmented lagrangian epoch.



5.5.1.4 Testing for the paradoxical effect

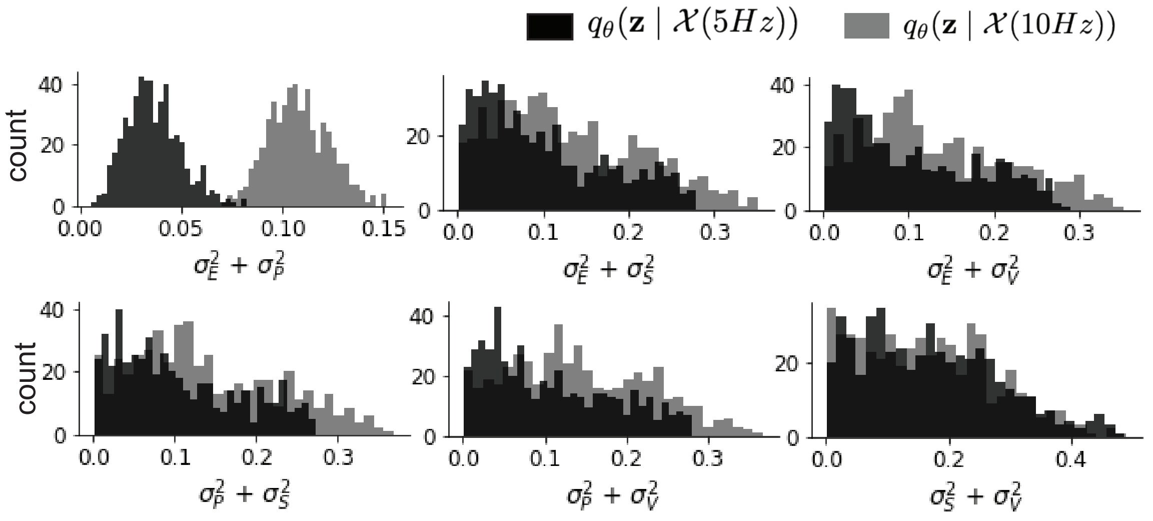
The paradoxical effect occurs when a populations steady state rate is decreased (or increased) when an increase (decrease) in current is applied to that population [170]. To see which, if any, populations exhibited a paradoxical effect, we examined responses to changes in input to individual neuron type populations, where the initial condition was the steady-state response to **h** (Fig. 5.7). Input magnitudes were chosen so that the effect is salient (0.002 for E and P, but 0.02 for S and V). Only the P-population exhibited the paradoxical effect at this connectivity W and input **h**.

5.5.1.5 Primary visual cortex: Mathematical intuition and challenges

We re-write the original Equations 5.7 and 5.8 in the following way:

$$\begin{aligned} d\mathbf{x} &= \frac{1}{\tau}(-\mathbf{x} + \phi(W\mathbf{x} + h + \epsilon))dt \\ d\epsilon &= -\frac{dt}{\tau_{\text{noise}}}\epsilon + \frac{\sqrt{2}}{\sqrt{\tau_{\text{noise}}}}\Sigma_\epsilon dW \end{aligned} \tag{5.20}$$

Figure 5.6: EPI predictive distributions of the sum of squares of each pair of noise parameters.



Where in this paper we chose Σ_ϵ, the covariance of the noise to be

$$\Sigma_\epsilon = \tau_{\text{noise}} \begin{bmatrix} \tilde{\sigma}_E & 0 & 0 & 0 \\ 0 & \tilde{\sigma}_P & 0 & 0 \\ 0 & 0 & \tilde{\sigma}_S & 0 \\ 0 & 0 & 0 & \tilde{\sigma}_V \end{bmatrix} \tag{5.21}$$

and where $\tilde{\sigma}_\alpha$ is the reparameterized standard deviation of the noise for population α from Equation 5.9.

We are interested in computing the covariance of the activity. For that, first we define $\mathbf{v} = \omega\mathbf{x}+\mathbf{h}+\epsilon$, the total input to each cell type, and the matrix S, the negative Jacobian $S = I-\omega f'(-\mathbf{v})$. Then, Eq. (5.20) can be written as an 8-dimensional system. Linearizing around the fixed point of the system without fluctuations, we find the equations that describe the fluctuations of the input to each cell type:

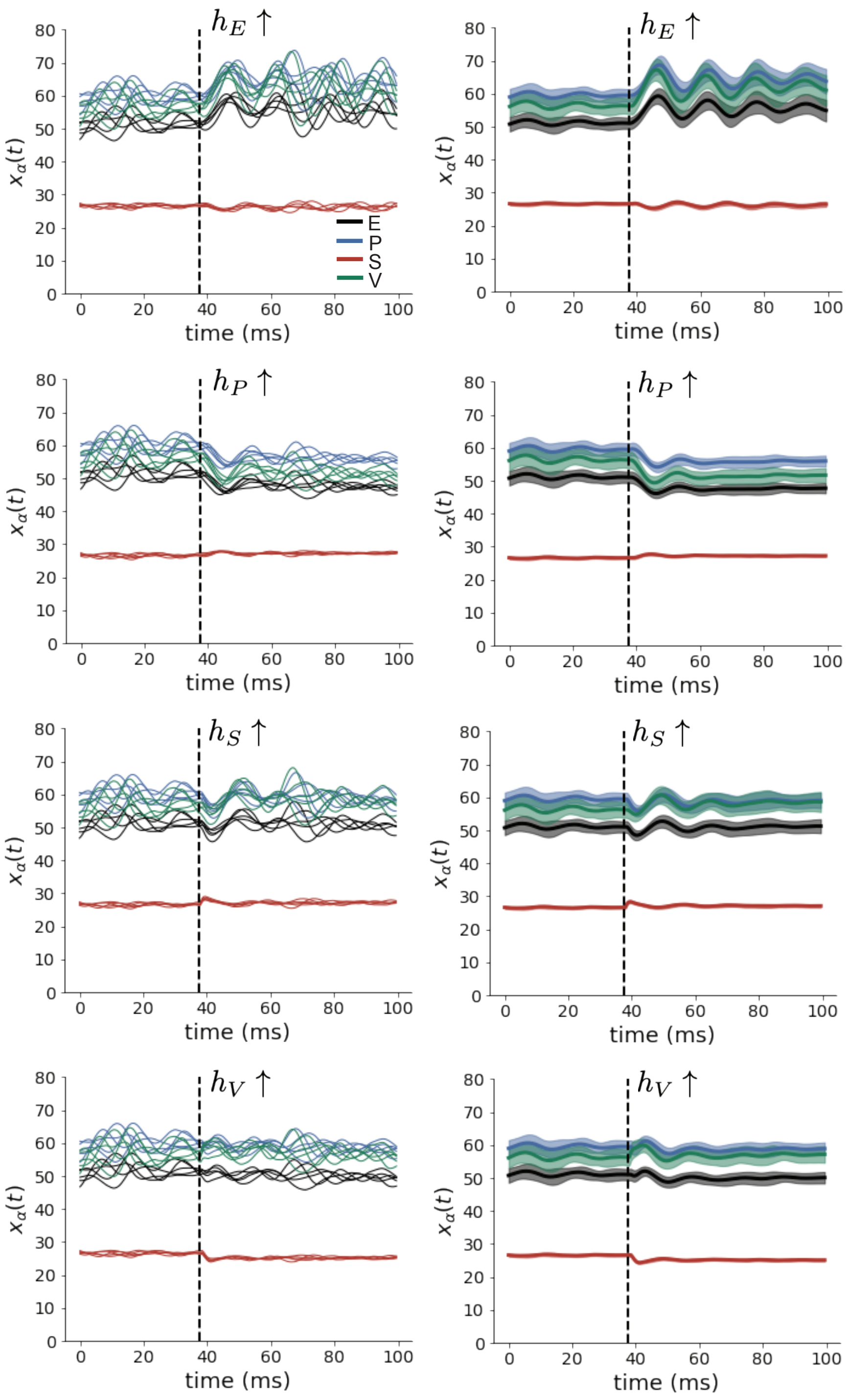


Figure 5.7: (Left) SSSN simulations for small increases in neuron-type population input. (Right) Average (solid) and standard deviation (shaded) of stochastic fluctuations of responses.

$$d\begin{pmatrix}\delta v\\ \epsilon\end{pmatrix} = -\begin{pmatrix}S & -\frac{\tau_{\text{noise}}-\tau}{\tau\tau_{\text{noise}}}I\\ 0 & \frac{1}{\tau_{\text{noise}}}I\end{pmatrix}\begin{pmatrix}\delta v\\ \epsilon\end{pmatrix}dt + \begin{pmatrix}0 & \frac{\sqrt{2}}{\sqrt{\tau_{\text{noise}}}}\Sigma_\epsilon\\ 0 & \frac{\sqrt{2}}{\sqrt{\tau_{\text{noise}}}}\Sigma_\epsilon\end{pmatrix}d\mathbf{W} \tag{5.22}$$

Where $d\mathbf{W}$ is a vector with the private noise of each variable. The $d\mathbf{W}$ term is multiplied by a non-diagonal matrix is because the noise that the voltage receives is the exact same than the one that comes from the OU process and not another process. The covariance of the inputs $\Lambda_v = \langle \delta\mathbf{v}\delta\mathbf{v}^T\rangle$ can be found as the solution the following Lyapunov Equation [228, 235]:

$$\begin{pmatrix}S & -\frac{\tau_{\text{noise}}-\tau}{\tau\tau_{\text{noise}}}I\\ 0 & \frac{1}{\tau_{\text{noise}}}I\end{pmatrix}\begin{pmatrix}\Lambda_v & \Lambda_c\\ \Lambda_c^T & \Lambda_\epsilon\end{pmatrix} + \begin{pmatrix}\Lambda_v & \Lambda_c\\ \Lambda_c^T & \Lambda_\epsilon\end{pmatrix}\begin{pmatrix}S^T & 0\\ -\frac{\tau_{\text{noise}}-\tau}{\tau\tau_{\text{noise}}}I & \frac{1}{\tau_{\text{noise}}}I\end{pmatrix} = \begin{pmatrix}\frac{2}{\tau_{\text{noise}}}\Lambda_\epsilon & \frac{2}{\tau_{\text{noise}}}\Lambda_\epsilon\\ \frac{2}{\tau_{\text{noise}}}\Lambda_\epsilon & \frac{2}{\tau_{\text{noise}}}\Lambda_\epsilon\end{pmatrix} \tag{5.23}$$

Where $\Lambda_c = \langle \delta\mathbf{v}\delta\epsilon^T\rangle$ can be eliminated by solving this block matrix multiplication:

$$S\Lambda_v + \Lambda_v S^T = \frac{2\Lambda_\epsilon}{\tau_{\text{noise}}} + \frac{\tau_{\text{noise}}^2 - \tau^2}{(\tau\tau_{\text{noise}})^2}\left((\frac{1}{\tau_{\text{noise}}}I + S)^{-1}\Lambda_\epsilon + \Lambda_\epsilon(\frac{1}{\tau_{\text{noise}}}I + S^T)^{-1}\right) \tag{5.24}$$

The equation above is another Lyapunov Equation, now in 4 dimensions. In the simplest case in which $\tau_{\text{noise}} = \tau$, the voltage is directly driven by white noise, and Λ_v can be expressed in powers of S and S^T. Because S satisfies its own polynomial equation (Cayley Hamilton theorem), there will be 4 coefficients for the expansion of S and 4 for S^T, resulting in 16 coefficients that define Λ_v for a given S. Due to symmetry arguments[235], in this case the diagonal elements of the covariance matrix of the voltage will have the form:

$$\Lambda_{v_{ii}} = \sum_{i=\{E,P,S,V\}} g_i(S)\sigma_{ii}^2 \tag{5.25}$$

These coefficients $g_i(S)$ are intricate functions of the Jacobian of the system. Although expressions for these coefficients can be found explicitly, only numerical evaluation of those expressions determine which components of the noisy input are going to strongly influence the variability of excitatory population. Showing the generality of this dependence in more complicated noise scenarios (e.g. $\tau_{\text{noise}} > \tau$ as in Section 5.1), is the focus of current research.

5.5.2 Superior colliculus

5.5.2.1 SC model

The ability to switch between two separate tasks throughout randomly interleaved trials, or “rapid task switching," has been studied in rats, and midbrain superior colliculus (SC) has been show to play an important in this computation [236]. Neural recordings in SC exhibited two populations of neurons that simultaneously represented both task context (Pro or Anti) and motor response (contralateral or ipsilateral to the recorded side), which led to the distinction of two functional classes: the Pro/Contra and Anti/Ipsi neurons [199]. Given this evidence, Duan et al. proposed a model with four functionally-defined neuron-type populations: two in each hemisphere corresponding to the Pro/Contra and Anti/Ipsi populations. We study how the connectivity of this neural circuit governs rapid task switching ability.

The four populations of this model are denoted as left Pro (LP), left Anti (LA), right Pro (RP) and right Anti (RA). Each unit has an activity (x_α) and internal variable (u_α) related by

$$x_\alpha = \phi(u_\alpha) = \left(\frac{1}{2}\tanh\left(\frac{u_\alpha - a}{b}\right) + \frac{1}{2}\right), \tag{5.26}$$

where $\alpha \in \{LP, LA, RA, RP\}$, $a = 0.05$ and $b = 0.5$ control the position and shape of the nonlinearity. We order the neural populations of x and u in the following manner

$$\mathbf{x} = \begin{bmatrix} x_{LP} \\ x_{LA} \\ x_{RP} \\ x_{RA} \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} u_{LP} \\ u_{LA} \\ u_{RP} \\ u_{RA} \end{bmatrix}, \tag{5.27}$$

which evolve according to

$$\tau \frac{d\mathbf{u}}{dt} = -\mathbf{u} + W\mathbf{x} + \mathbf{h} + d\mathbf{B}. \tag{5.28}$$

with time constant $\tau = 0.09s$, step size 24ms and Gaussian noise $d\mathbf{B}$ of variance 0.2^2. These hyperparameter values are motivated by modeling choices and results from [199].

The weight matrix has 4 parameters for self sW, vertical vW, horizontal hW, and diagonal dW connections:

$$W = \begin{bmatrix} sW & vW & hW & dW \\ vW & sW & dW & hW \\ hW & dW & sW & vW \\ dW & hW & vW & sW \end{bmatrix}. \tag{5.29}$$

We study the role of parameters $\mathbf{z} = [sW, vW, hW, dW]^\top$ in rapid task switching.

The circuit receives four different inputs throughout each trial, which has a total length of 1.8s.

$$\mathbf{h} = \mathbf{h}_{\text{constant}} + \mathbf{h}_{\text{P,bias}} + \mathbf{h}_{\text{rule}} + \mathbf{h}_{\text{choice-period}} + \mathbf{h}_{\text{light}}. \tag{5.30}$$

There is a constant input to every population,

$$\mathbf{h}_{\text{constant}} = I_{\text{constant}}[1, 1, 1, 1]^\top, \tag{5.31}$$

a bias to the Pro populations

$$\mathbf{h}_{\text{P,bias}} = I_{\text{P,bias}}[1, 0, 1, 0]^\top, \tag{5.32}$$

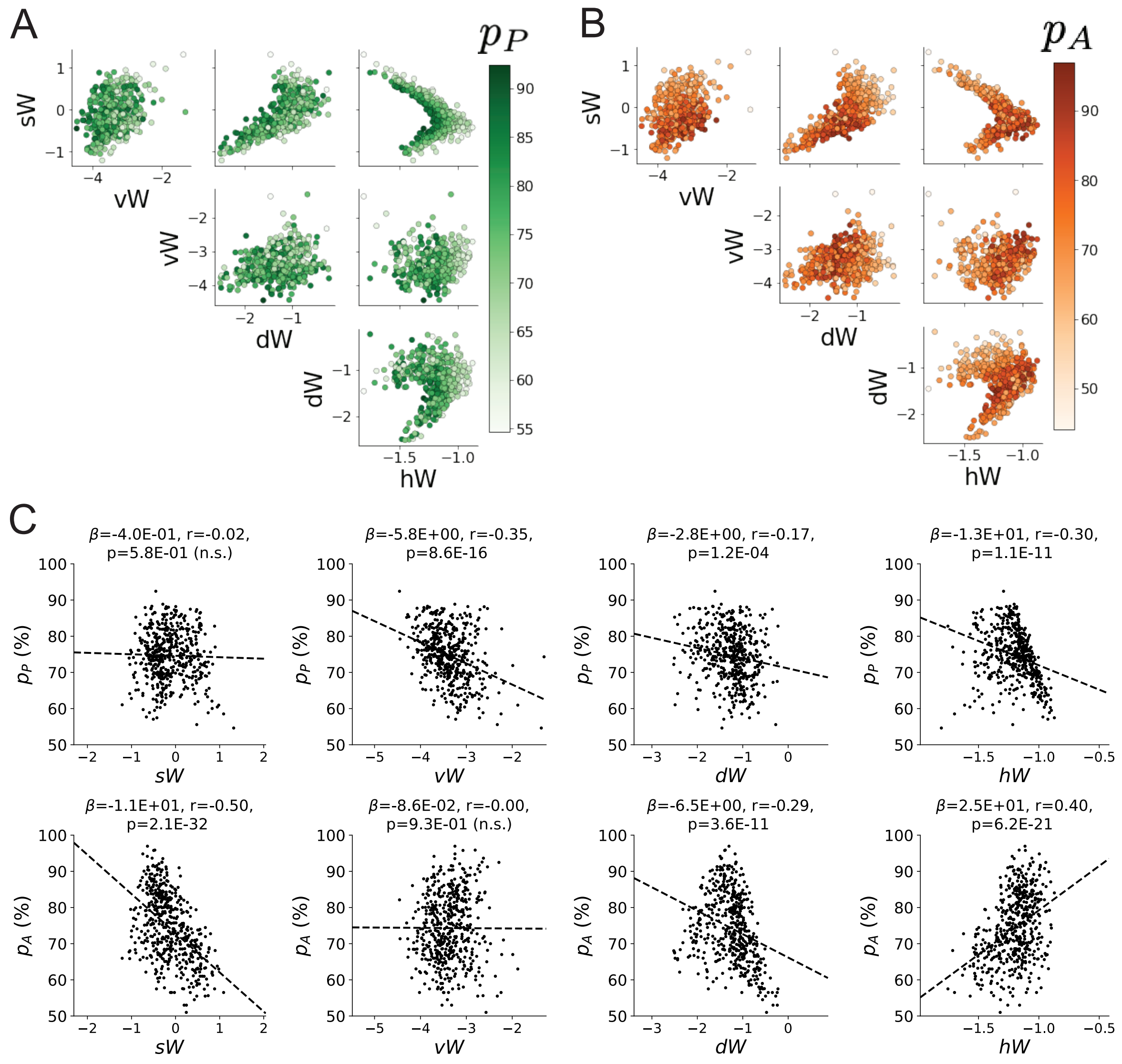


Figure 5.8: **A**. Same pairplot as Fig. 5.2C colored by Pro task accuracy. **B**. Same as A colored by Anti task accuracy. **C**. Connectivity parameters of EPI distributions versus task accuracies. β is slope coefficient of linear regression, r is correlation, and p is the two-tailed p-value.

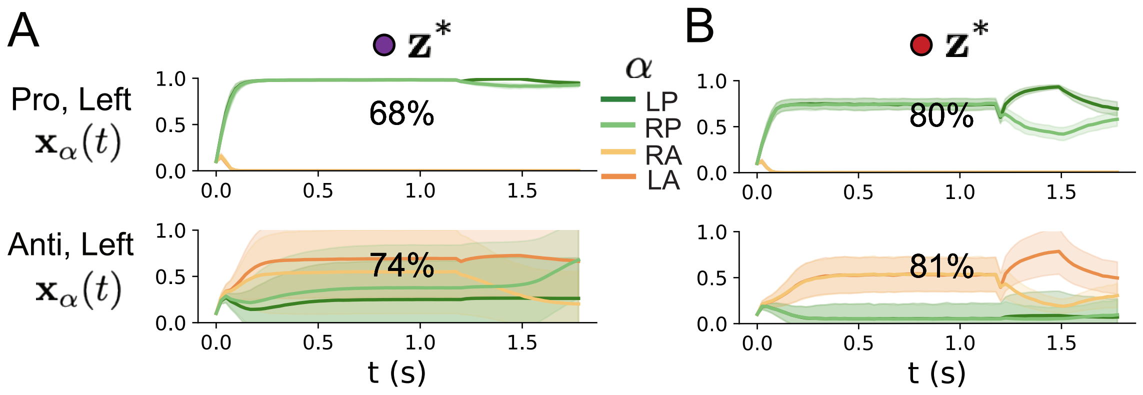


Figure 5.9: **A**. Simulations in network regime 1: $\mathbf{z}^*(sW = -0.75)$. **B**. Simulations in network regime 2: $\mathbf{z}^*(sW = 0.75)$.

rule-based input depending on the condition

$$\mathbf{h}_{\text{P,rule}}(t) = \begin{cases} I_{\text{P,rule}}[1, 0, 1, 0]^\top, & \text{if } t \leq 1.2s \\ 0, & \text{otherwise} \end{cases} \tag{5.33}$$

$$\mathbf{h}_{\text{A,rule}}(t) = \begin{cases} I_{\text{A,rule}}[0, 1, 0, 1]^\top, & \text{if } t \leq 1.2s \\ 0, & \text{otherwise} \end{cases}, \tag{5.34}$$

a choice-period input

$$\mathbf{h}_{\text{choice}}(t) = \begin{cases} I_{\text{choice}}[1, 1, 1, 1]^\top, & \text{if } t > 1.2s \\ 0, & \text{otherwise} \end{cases}, \tag{5.35}$$

and an input to the right or left-side depending on where the light stimulus is delivered

$$\mathbf{h}_{\text{light}}(t) = \begin{cases} I_{\text{light}}[1, 1, 0, 0]^\top, & \text{if } 1.2s < t < 1.5s \text{ and Left} \\ I_{\text{light}}[0, 0, 1, 1]^\top, & \text{if } 1.2s < t < 1.5s \text{ and Right} \\ 0, & \text{otherwise} \end{cases}. \tag{5.36}$$

The input parameterization was fixed to $I_{\text{constant}} = 0.75$, $I_{\text{P,bias}} = 0.5$, $I_{\text{P,rule}} = 0.6$, $I_{\text{A,rule}} = 0.6$,

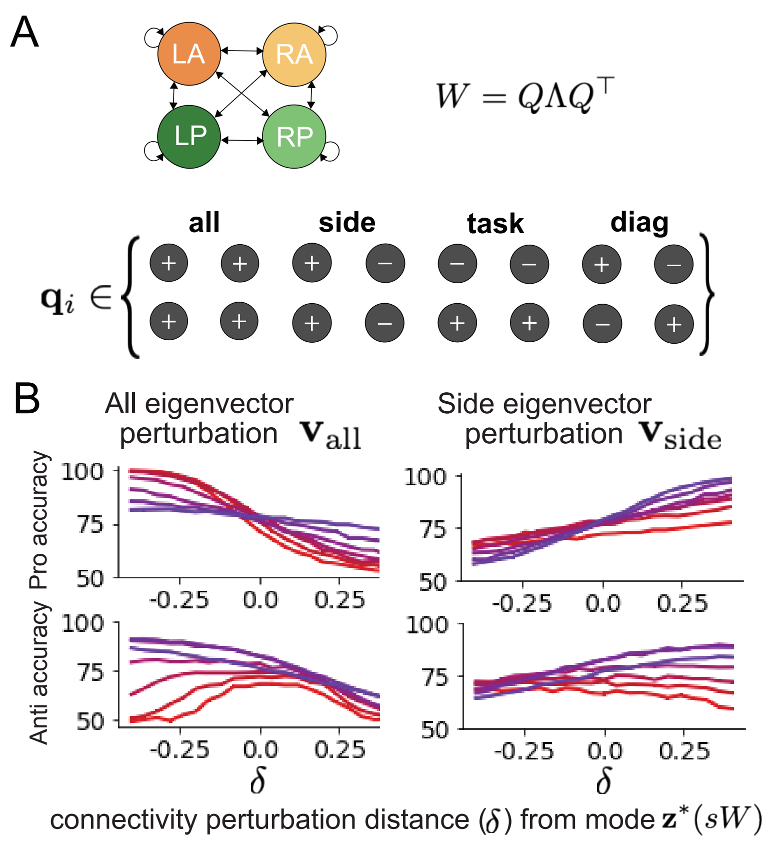


Figure 5.10: **A**. Invariant eigenvectors of connectivity matrix W. **B**. Accuracies for connectivity perturbations when changing λ_{all} and λ_{side} (λ_{task} and λ_{diag} shown in Fig. 5.2D).

$I_{\text{choice}} = 0.25$, and $I_{\text{light}} = 0.5$.

5.5.2.2 Task accuracy calculation

The accuracies of the Pro and Anti tasks are calculated as

$$p_P(\mathbf{x}; \mathbf{z}) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})}\left[d_P(\mathbf{x}; \mathbf{z})\right] \tag{5.37}$$

and

$$p_A(\mathbf{x}; \mathbf{z}) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})}\left[d_A(\mathbf{x}; \mathbf{z})\right] \tag{5.38}$$

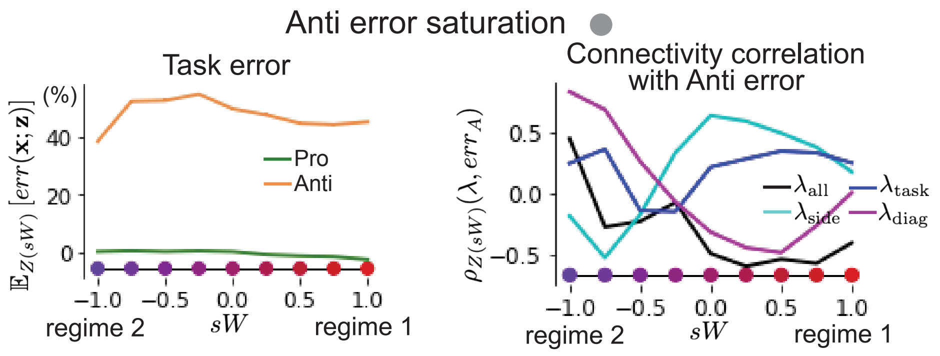


Figure 5.11: (Left) Mean and standard error of Pro and Anti error from regime 1 to regime 2 at $\gamma = 0.85$. (Right) Correlations of connectivity eigenvalues with Anti error from regime 1 to regime 2 at $\gamma = 0.85$.

where $d_P(\mathbf{x};\mathbf{z})$ and $d_A(\mathbf{x};\mathbf{z})$ calculate the decision made in each trial (approximately 1 for correct and 0 for incorrect choices). Specifically,

$$d_P(\mathbf{x};\mathbf{z}) = \Theta[x_{LP}(t = 1.8s) - x_{RP}(t = 1.8s)] \tag{5.39}$$

in Pro trials where the stimulus is on the left side, and Θ approximates the Heaviside step function. Similarly,

$$d_A(\mathbf{x};\mathbf{z}) = \Theta[x_{RP}(t = 1.8s) - x_{LP}(t = 1.8s)] \tag{5.40}$$

in Anti trials where the stimulus was on the left side. Our accuracy calculation only considers one stimulus presentation (Left), since the model is left-right symmetric. The accuracy is averaged over 200 independent trials, and the Heaviside step function is approximated as

$$\Theta(\mathbf{x}) = \text{sigmoid}(\beta_\Theta \mathbf{x}), \tag{5.41}$$

where $\beta_\Theta = 100$.

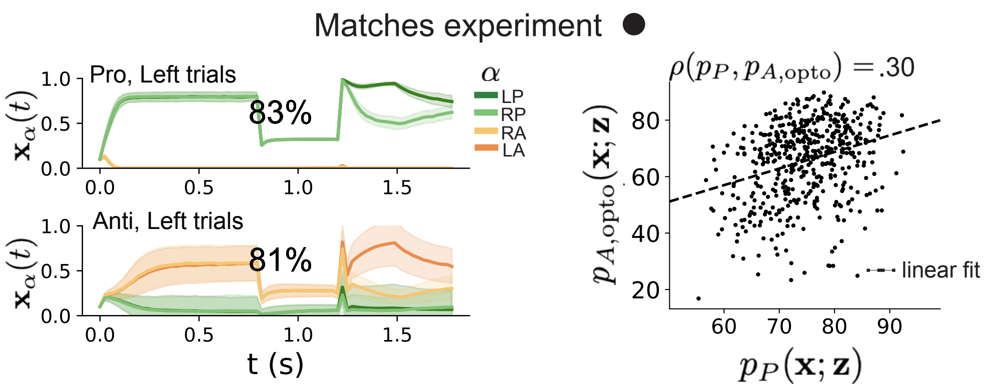


Figure 5.12: (Left) Mean and standard deviation (shading) of responses of the SC model at the mode of the EPI distribution to delay period inactivation at $\gamma = 0.675$. Accuracy in Pro (top) and Anti (bottom) task is shown as a percentage. (Right) Anti accuracy following delay period inactivation at $\gamma = 0.675$ versus accuracy in the Pro task across connectivities in the EPI distribution.

5.5.2.3 EPI details for the SC model

To write the emergent properties of Equation 5.4 in terms of the EPI optimization, we have

$$f(\mathbf{x};\mathbf{z}) = \begin{bmatrix} d_P(\mathbf{x};\mathbf{z}) \\ d_A(\mathbf{x};\mathbf{z}) \end{bmatrix} \tag{5.42}$$

$$\boldsymbol{\mu} = \begin{bmatrix} .75 \\ .75 \end{bmatrix}, \tag{5.43}$$

and

$$\boldsymbol{\sigma}^2 = \begin{bmatrix} .075^2 \\ .075^2 \end{bmatrix} \tag{5.44}$$

(see Sections 4.4.1.3-4.4.1.4, and example in Section 4.4.1.5).

Throughout optimization, the augmented lagrangian parameters η and c, were updated after each epoch of $i_{\text{max}} = 2,000$ iterations (see Section 4.4.1.4). The optimization converged after ten epochs (Fig. 5.12).

For EPI in Fig. 5.2C, we used a real NVP architecture with three coupling layers of affine

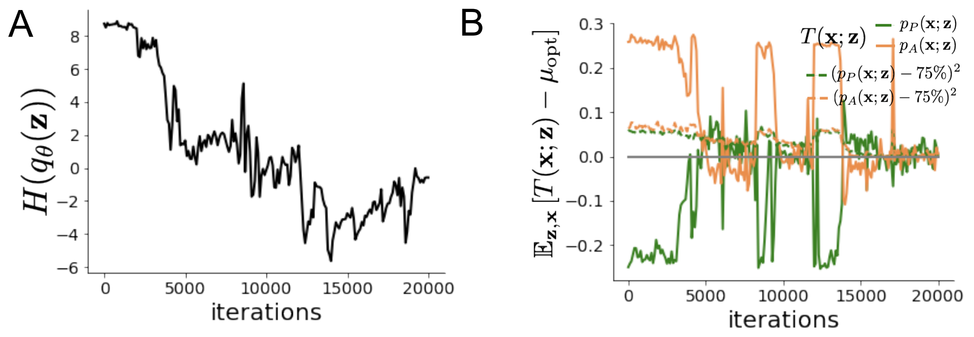


Figure 5.13: EPI optimization of the SC model producing rapid task switching. **A**. Entropy throughout optimization. **B**. The emergent property statistic means and variances converge to their constraints at 20,000 iterations following the tenth augmented lagrangian epoch.

transformations parameterized by two-layer neural networks of 50 units per layer. The initial distribution was a standard isotropic gaussian $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, I)$ mapped to a support of $\mathbf{z}_i \in [-5, 5]$. We used an augmented lagrangian coefficient of $c_0 = 10^2$, a batch size $n = 100$, and $\beta = 2$. The distribution was the greatest EPI distribution to converge across 5 random seeds with criteria $N_{\text{test}} = 25$.

The bend in the EPI distribution is not a spurious result of the EPI optimization. The structure discovered by EPI matches the shape of the set of points returned from brute-force random sampling (Fig. 5.14A) These connectivities were sampled from a uniform distribution over the range of each connectivity parameter, and all parameters producing accuracy in each task within the range of 60% to 90% were kept. This set of connectivities will not match the distribution of EPI exactly, since it is not conditioned on the emergent property. For example the parameter set returned by the brute-force search is biased towards lower accuracies (Fig. 5.14B).

5.5.2.4 Mode identification with EPI

We found one mode of the EPI distribution for fixed values of sW from 1 to -1 in steps of 0.25. To begin, we chose an initial parameter value from 500 parameter samples $\mathbf{z} \sim q_\theta(\mathbf{z} \mid \mathcal{X})$ that had closest sW value to 1. We then optimized this estimate of the mode (for fixed sW) using probability gradients of the deep probability distribution for 500 steps of gradient ascent with a

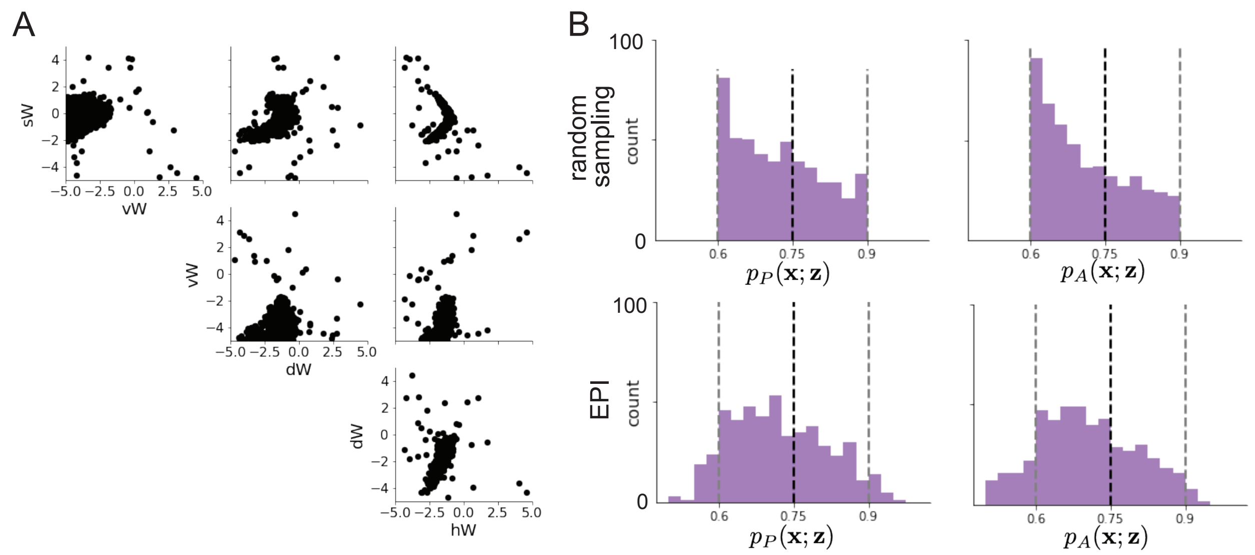


Figure 5.14: **A**.Rapid task switching SC connectivities obtained from random sampling. **B**. Task accuracies of the inferred distributions from random sampling (top) and EPI (bottom).

learning rate of 5×10^{-3}. The next mode (at $sW = 0.75$) was found using the previous mode as the initialization. This and all subsequent optimizations used 200 steps of gradient ascent with a learning rate of 1×10^{-3}, except at $sW = -1$ where a learning rate of 5×10^{-4} was used. During all mode identification optimizations, the learning rate was reduced by half (decay = 0.5) after every 100 iterations.

5.5.2.5 Sample grouping by mode

For the analyses in Figure 5.3C and Figure 5.11, we obtained parameters for each step along the continuum between regimes 1 and 2 by sampling from the EPI distribution. Each sample was assigned to the closest mode $\mathbf{z}^*(sW)$. Sampling continued until 500 samples were assigned to each mode, which took 2.67 seconds (5.34ms/sample-per-mode). It took 9.59 minutes to obtain just 5 samples for each mode with brute force sampling requiring accuracies between 60% and 90% in each task (115s/sample-per-mode). This corresponds to a sampling speed increase of roughly 21,500 once the EPI distribution has been learned.

5.5.2.6 Sensitivity analysis

At each mode, we measure the sensitivity dimension (that of most negative eigenvalue in the Hessian of the EPI distribution) $\mathbf{v}_1(\mathbf{z}^*)$. To resolve sign degeneracy in eigenvectors, we chose $\mathbf{v}_1(\mathbf{z}^*)$ to have negative element in hW. This tells us what parameter combination rapid task switching is most sensitive to at this parameter choice in the regime.

5.5.2.7 Connectivity eigendecomposition and processing modes

To understand the connectivity mechanisms governing task accuracy, we took the eigendecomposition of the connectivity matrices $W = Q\Lambda Q^{-1}$, which results in the same eigenmodes $\mathbf{q}_i$ for all W parameterized by $\mathbf{z}$ (Fig. 5.10A). These eigenvectors are always the same, because the connectivity matrix is symmetric and the model also assumes symmetry across hemispheres, but the eigenvalues of connectivity (or degree of eigenmode amplification) change with $\mathbf{z}$. These basis vectors have intuitive roles in processing for this task, and are accordingly named the *all* eigenmode - all neurons co-fluctuate, *side* eigenmode - one side dominates the other, *task* eigenmode - the Pro or Anti populations dominate the other, and *diag* mode - Pro- and Anti-populations of opposite hemispheres dominate the opposite pair. Due to the parametric structure of the connectivity matrix, the parameters $\mathbf{z}$ are a linear function of the eigenvalues $\boldsymbol{\lambda} = [\lambda_{\text{all}}, \lambda_{\text{side}}, \lambda_{\text{task}} \lambda_{\text{diag}}]^\top$ associated with these eigenmodes.

$$\mathbf{z} = A\boldsymbol{\lambda} \tag{5.45}$$

$$A = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \end{bmatrix}. \tag{5.46}$$

We are interested in the effect of raising or lowering the amplification of each eigenmode in the connectivity matrix by perturbing individual eigenvalues λ. To test this, we calculate the unit vector of changes in the connectivity $\mathbf{z}$ that result from a change in the associated eigenvalues

$$\mathbf{v}_{\mathrm{a}} = \frac{\frac{\partial \mathbf{z}}{\partial \lambda_{\mathrm{a}}}}{|\frac{\partial \mathbf{z}}{\partial \lambda_{\mathrm{a}}}|_2}, \tag{5.47}$$

where

$$\frac{\partial \mathbf{z}}{\partial \lambda_{\mathrm{a}}} = A\mathbf{e}_{\mathrm{a}}, \tag{5.48}$$

and e.g. $\mathbf{e}_{\mathrm{all}} = [1, 0, 0, 0]^\top$. So $\mathbf{v}_{\mathrm{a}}$ is the normalized column of A corresponding to eigenmode a. The parameter dimension $\mathbf{v}_a$ ($a \in \{\mathrm{all}, \mathrm{side}, \mathrm{task}, \text{ and diag}\}$) that increases the eigenvalue of connectivity λ_a is $\mathbf{z}$-invariant (Equation 5.48) and $\mathbf{v}_a \perp \mathbf{v}_{b \neq a}$. By perturbing $\mathbf{z}$ along $\mathbf{v}_a$, we can examine how model function changes by directly modulating the connectivity amplification of specific eigenmodes, which have interpretable roles in processing in each task.

5.5.2.8 Modeling optogenetic silencing.

We tested whether the inferred SC model connectivities could reproduce experimental effects of optogenetic inactivation in rats [199]. During periods of simulated optogenetic inactivation, activity was decreased proportional to the optogenetic strength $\gamma \in [0, 1]$

$$x_\alpha = (1 - \gamma)\phi(u_\alpha). \tag{5.49}$$

Delay period inactivation was from $0.8 < t < 1.2$.

Conclusion

This dissertation focused on the development of machine learning techniques for theoretical neuroscience. For building normative theories, we developed optimization techniques for predicting neural responses that encode muscle activity according to different normative criteria (Chapter 2). For building descriptive theories, we developed a technique for efficient inference in statistical generative models of neural data that belong to the exponential family (Chapter 3). For building mechanistic theories, we developed a parameter inference technique for neural circuit models that constrains the inferred parameter distribution to produce an emergent property of computation (Chapter 4). Emergent property inference was used for novel insight regarding mechanistic models of primary visual cortex and superior colliculus (Chapter 5).

The methods introduced in Chapters 3 and 4 used normalizing flows to create powerful deep generative modeling techniques. By fitting distributions (defined by deep neural networks) to complex parameter distributions, we were then able to quantify and characterize the scientifically meaningful structure captured by these deep generative models. In Chapter 5, we showed how effectively such techniques can be used for scientific analysis, and believe this to be a powerful approach to be used in future theoretical work.

References

[1] Eric R Kandel et al. *Principles of neural science*. Vol. 4. McGraw-hill New York, 2000.

[2] Mark Bear, Barry Connors, and Michael A Paradiso. *Neuroscience: Exploring the brain*. Jones & Bartlett Learning, LLC, 2020.

[3] Peter Dayan and Laurence F Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Computational Neuroscience Series, 2001.

[4] Eugene M Izhikevich. *Dynamical systems in neuroscience*. MIT press, 2007.

[5] Wulfram Gerstner et al. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[6] Christof Koch and Idan Segev. “The role of single neurons in information processing”. In: *Nature neuroscience* 3.11 (2000), pp. 1171–1177.

[7] Michael London and Michael Häusser. “Dendritic computation”. In: *Annu. Rev. Neurosci.* 28 (2005), pp. 503–532.

[8] Nancy Kopell and G Bard Ermentrout. “Coupled oscillators and the design of central pattern generators”. In: *Mathematical biosciences* 90.1-2 (1988), pp. 87–109.

[9] Eve Marder. “From biophysics to models of network function”. In: *Annual review of neuroscience* 21.1 (1998), pp. 25–45.

[10] Larry F Abbott. “Theoretical neuroscience rising”. In: *Neuron* 60.3 (2008), pp. 489–495.

[11] Xiao-Jing Wang. “Neurophysiological and computational principles of cortical rhythms in cognition”. In: *Physiological reviews* 90.3 (2010), pp. 1195–1268.

[12] Timothy O’Leary, Alexander C Sutton, and Eve Marder. “Computational models in the age of large datasets”. In: *Current opinion in neurobiology* 32 (2015), pp. 87–94.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[14] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. “Speech recognition with deep recurrent neural networks”. In: *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee. 2013, pp. 6645–6649.

[15] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[16] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

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

[18] Blake A Richards and et al. "A deep learning framework for neuroscience". In: *Nature Neuroscience* (2019).

[19] Daniel Levenstein et al. "On the role of theory and modeling in neuroscience". In: *arXiv preprint arXiv:2003.13825* (2020).

[20] David H Hubel and Torsten N Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex". In: *The Journal of physiology* 160.1 (1962), pp. 106–154.

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

[22] John O'keefe and Lynn Nadel. *The hippocampus as a cognitive map*. Oxford: Clarendon Press, 1978.

[23] Robert E Kass, Uri T Eden, and Emery N Brown. *Analysis of neural data*. Vol. 491. Springer, 2014.

[24] Robert E Kass and Valérie Ventura. "A spike-train probability model". In: *Neural computation* 13.8 (2001), pp. 1713–1720.

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

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

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

[28] Anne C Smith and Emery N Brown. "Estimating a state-space model from point process observations". In: *Neural computation* 15.5 (2003), pp. 965–991.

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

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

[31] Shreya Saxena and John P Cunningham. “Towards the neural population doctrine”. In: *Current opinion in neurobiology* 55 (2019), pp. 103–111.

[32] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *International Conference on Learning Representations* (2014).

[33] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. “Stochastic backpropagation and approximate inference in deep generative models”. In: *arXiv preprint arXiv:1401.4082* (2014).

[34] Yuanjun Gao et al. “Linear dynamical neural population models through nonlinear embeddings”. In: *NIPS*. 2016, pp. 163–171.

[35] Yuan Zhao and Il Memming Park. “Recursive variational bayesian dual estimation for nonlinear dynamics and non-gaussian observations”. In: *stat* 1050 (2017), p. 27.

[36] Gabriel Barello, Adam Charles, and Jonathan Pillow. “Sparse-Coding Variational Auto-Encoders”. In: *bioRxiv* (2018), p. 399246.

[37] Chethan Pandarinath et al. “Inferring single-trial neural population dynamics using sequential auto-encoders”. In: *Nature methods* (2018), p. 1.

[38] Alexander B Wiltschko et al. “Mapping sub-second structure in mouse behavior”. In: *Neuron* 88.6 (2015), pp. 1121–1135.

[39] Matthew J Johnson et al. “Composing graphical models with neural networks for structured representations and fast inference”. In: *Advances in neural information processing systems*. 2016, pp. 2946–2954.

[40] Eleanor Batty et al. “BehaveNet: nonlinear embedding and Bayesian neural decoding of behavioral videos”. In: *Advances in Neural Information Processing Systems* (2019).

[41] Gabrielle J Gutierrez, Timothy O’Leary, and Eve Marder. “Multiple mechanisms switch an electrically coupled, synaptically inhibited neuron between competing rhythmic oscillators”. In: *Neuron* 77.5 (2013), pp. 845–858.

[42] Shin-ya Takemura et al. “The comprehensive connectome of a neural substrate for ‘ON’ motion detection in Drosophila”. In: *Elife* 6 (2017), e24394.

[43] David Sussillo. “Neural circuits as computational dynamical systems”. In: *Current opinion in neurobiology* 25 (2014), pp. 156–163.

[44] Romain Brette et al. “Simulation of networks of spiking neurons: a review of tools and strategies”. In: *Journal of computational neuroscience* 23.3 (2007), pp. 349–398.

[45] Eve Marder and Vatsala Thirumalai. “Cellular, synaptic and network effects of neuromodulation”. In: *Neural Networks* 15.4-6 (2002), pp. 479–493.

[46] Omri Barak. “Recurrent neural networks as versatile tools of neuroscience research”. In: *Current opinion in neurobiology* 46 (2017), pp. 1–6.

[47] Xiao-Jing Wang et al. “Brain connectomes come of age”. In: *Current Opinion in Neurobiology* 65 (2020), pp. 152–161.

[48] Catherine Morris and Harold Lecar. “Voltage oscillations in the barnacle giant muscle fiber”. In: *Biophysical journal* 35.1 (1981), pp. 193–213.

[49] Helen H Yang and Thomas R Clandinin. “Elementary motion detection in Drosophila: algorithms and mechanisms”. In: *Annual Review of Vision Science* 4 (2018), pp. 143–163.

[50] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. “The frontier of simulation-based inference”. In: *Proceedings of the National Academy of Sciences* (2020).

[51] Mark A Beaumont, Wenyang Zhang, and David J Balding. “Approximate Bayesian computation in population genetics”. In: *Genetics* 162.4 (2002), pp. 2025–2035.

[52] Scott A Sisson, Yanan Fan, and Mark M Tanaka. “Sequential monte carlo without likelihoods”. In: *Proceedings of the National Academy of Sciences* 104.6 (2007), pp. 1760–1765.

[53] Donald Olding Hebb. “The organization of behavior; a neuropsycholocigal theory”. In: *A Wiley Book in Clinical Psychology* 62 (1949), p. 78.

[54] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.

[55] Paul J Werbos. “Applications of advances in nonlinear sensitivity analysis”. In: *System modeling and optimization*. Springer, 1982, pp. 762–770.

[56] Yann LeCun. “Une procedure d’apprentissage ponr reseau a seuil asymetrique”. In: *Proceedings of Cognitiva 85* (1985), pp. 599–604.

[57] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[58] David Sussillo and Omri Barak. "Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks". In: *Neural computation* 25.3 (2013), pp. 626–649.

[59] Paul Werbos. "Beyond regression:" new tools for prediction and analysis in the behavioral sciences". In: *Ph. D. dissertation, Harvard University* (1974).

[60] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[61] Herbert Jaeger and Harald Haas. "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication". In: *science* 304.5667 (2004), pp. 78–80.

[62] David Sussillo and Larry F Abbott. "Generating coherent patterns of activity from chaotic neural networks". In: *Neuron* 63.4 (2009), pp. 544–557.

[63] Timothy P Lillicrap et al. "Random synaptic feedback weights support error backpropagation for deep learning". In: *Nature communications* 7.1 (2016), pp. 1–10.

[64] James M Murray. "Local online learning in recurrent networks with random feedback". In: *ELife* 8 (2019), e43299.

[65] Daniel LK Yamins et al. "Performance-optimized hierarchical models predict neural responses in higher visual cortex". In: *Proceedings of the national academy of sciences* 111.23 (2014), pp. 8619–8624.

[66] Bruno A Olshausen and David J Field. "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381.6583 (1996), pp. 607–609.

[67] Ian J Goodfellow et al. "Generative adversarial networks". In: *arXiv preprint arXiv:1406.2661* (2014).

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

[69] George Papamakarios et al. "Normalizing flows for probabilistic modeling and inference". In: *arXiv preprint arXiv:1912.02762* (2019).

[70] Durk P Kingma et al. "Improved variational inference with inverse autoregressive flow". In: *Advances in neural information processing systems* 29 (2016), pp. 4743–4751.

[71] George Papamakarios, Iain Murray, and Theo Pavlakou. “Masked autoregressive flow for density estimation”. In: *NIPS*. 2017, pp. 2335–2344.

[72] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. “Density estimation using real nvp”. In: *Proceedings of the 5th International Conference on Learning Representations* (2017).

[73] Abigail A Russo et al. “Motor cortex embeds muscle-like commands in an untangled population response”. In: *Neuron* 97.4 (2018), pp. 953–966.

[74] Abigail A Russo et al. “Neural trajectories in the supplementary motor area and motor cortex exhibit distinct geometries, compatible with different classes of computation”. In: *Neuron* 107.4 (2020), pp. 745–758.

[75] Richard P Dum and Peter L Strick. “The origin of corticospinal projections from the premotor areas in the frontal lobe”. In: *Journal of Neuroscience* 11.3 (1991), pp. 667–689.

[76] Jean-Alban Rathelot and Peter L Strick. “Subdivisions of primary motor cortex based on cortico-motoneuronal cells”. In: *Proceedings of the National Academy of Sciences* 106.3 (2009), pp. 918–923.

[77] Andrew B Schwartz. “Direct cortical representation of drawing”. In: *Science* 265.5171 (1994), pp. 540–542.

[78] Daniel W Moran and Andrew B Schwartz. “Motor cortical activity during drawing movements: population representation during spiral tracing”. In: *Journal of neurophysiology* 82.5 (1999), pp. 2693–2704.

[79] Emanuel Todorov. “Direct cortical control of muscle activation in voluntary arm movements: a model”. In: *Nature neuroscience* 3.4 (2000), pp. 391–398.

[80] Lauren E Sergio, Catherine Hamel-Pâquet, and John F Kalaska. “Motor cortex neural correlates of output kinematics and kinetics during isometric-force and arm-reaching tasks”. In: *Journal of neurophysiology* 94.4 (2005), pp. 2353–2378.

[81] Robert Ajemian et al. “Assessing the function of motor cortex: single-neuron models of how neural response is modulated by limb biomechanics”. In: *Neuron* 58.3 (2008), pp. 414–428.

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

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

[84] Thomas M Hall, Felipe de Carvalho, and Andrew Jackson. “A common structure underlies low-frequency cortical dynamics in movement, sleep, and sedation”. In: *Neuron* 83.5 (2014), pp. 1185–1199.

[85] David Sussillo et al. “A neural network that finds a naturalistic solution for the production of muscle activity”. In: *Nature neuroscience* 18.7 (2015), pp. 1025–1033.

[86] Jonathan A Michaels, Benjamin Dann, and Hansjörg Scherberger. “Neural population dynamics during reaching are better explained by a dynamical system than representational tuning”. In: *PLoS computational biology* 12.11 (2016), e1005175.

[87] Jeffrey S Seely et al. “Tensor analysis reveals distinct population structure that parallels the different computational roles of areas M1 and V1”. In: *PLoS computational biology* 12.11 (2016), e1005164.

[88] Shaul Druckmann and Dmitri B Chklovskii. “Neuronal circuits underlying persistent representations despite time varying activity”. In: *Current Biology* 22.22 (2012), pp. 2095–2103.

[89] Matthew T Kaufman et al. “Cortical activity in the null space: permitting preparation without movement”. In: *Nature neuroscience* 17.3 (2014), pp. 440–448.

[90] Darcy Michelle Griffin et al. “Do corticomotoneuronal cells predict target muscle EMG activity?” In: *Journal of neurophysiology* 99.3 (2008), pp. 1169–1986.

[91] Michelle M Morrow, Eric A Pohlmeyer, and Lee E Miller. “Control of muscle synergies by cortical ensembles”. In: *Progress in Motor Control*. Springer, 2009, pp. 179–199.

[92] Marc H Schieber and Gil Rivlis. “Partial reconstruction of muscle activity from a pruned network of diverse motor cortex neurons”. In: *Journal of neurophysiology* 97.1 (2007), pp. 70–82.

[93] Edward V Evarts. “Relation of pyramidal tract activity to force exerted during voluntary movement.” In: *Journal of neurophysiology* 31.1 (1968), pp. 14–27.

[94] Jaime Cadena-Valencia et al. “Entrainment and maintenance of an internal metronome in supplementary motor area”. In: *Elife* 7 (2018), e38983.

[95] Katja Kornysheva and Jörn Diedrichsen. “Human premotor areas parse sequences into their spatial and temporal features”. In: *Elife* 3 (2014), e03043.

[96] Hugo Merchant and Victor De Lafuente. “Introduction to the neurobiology of interval timing”. In: *Neurobiology of interval timing* (2014), pp. 1–13.

[97] Hajime Mushiake, Masahiko Inase, and Jun Tanji. "Neuronal activity in the primate premotor, supplementary, and precentral motor cortex during visually guided and internally determined sequential movements". In: *Journal of neurophysiology* 66.3 (1991), pp. 705–718.

[98] Kae Nakamura, Katsuyuki Sakai, and Okihide Hikosaka. "Neuronal activity in medial frontal cortex during learning of sequential procedures". In: *Journal of neurophysiology* 80.5 (1998), pp. 2671–2687.

[99] Evan D Remington et al. "Flexible sensorimotor computations through rapid reconfiguration of cortical dynamics". In: *Neuron* 98.5 (2018), pp. 1005–1019.

[100] Wolfram Schultz and Ranulfo Romo. "Role of primate basal ganglia and frontal cortex in the internal generation of movements". In: *Experimental Brain Research* 91.3 (1992), pp. 363–384.

[101] Keisetsu Shima and Jun Tanji. "Neuronal activity in the supplementary and presupplementary motor areas for temporal organization of multiple movements". In: *Journal of neurophysiology* 84.4 (2000), pp. 2148–2160.

[102] Jeong-Woo Sohn and Daeyeol Lee. "Order-dependent modulation of directional signals in the supplementary and presupplementary motor areas". In: *Journal of Neuroscience* 27.50 (2007), pp. 13655–13666.

[103] J Tanji and KIYOSHI Kurata. "Comparison of movement-related activity in two cortical motor areas of primates." In: *Journal of Neurophysiology* 48.3 (1982), pp. 633–653.

[104] Jun Tanji and Keisetsu Shima. "Role for supplementary motor area cells in planning several movements ahead". In: *Nature* 371.6496 (1994), pp. 413–416.

[105] D Thaler et al. "The functions of the medial premotor cortex". In: *Experimental Brain Research* 102.3 (1995), pp. 445–460.

[106] Jing Wang et al. "Flexible timing by temporal scaling of cortical responses". In: *Nature neuroscience* 21.1 (2018), p. 102.

[107] Laura N Driscoll, Matthew D Golub, and David Sussillo. "Computation through cortical dynamics". In: *Neuron* 98.5 (2018), pp. 873–875.

[108] Juan A Gallego et al. "Neural manifolds for the control of movement". In: *Neuron* 94.5 (2017), pp. 978–984.

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

[110] Mark Stopfer and Gilles Laurent. "Short-term memory in olfactory network dynamics". In: *Nature* 402.6762 (1999), pp. 664–668.

[111] James J DiCarlo and David D Cox. "Untangling invariant object recognition". In: *Trends in cognitive sciences* 11.8 (2007), pp. 333–341.

[112] Marino Pagan et al. "Signals in inferotemporal and perirhinal cortex suggest an untangling of visual target information". In: *Nature neuroscience* 16.8 (2013), pp. 1132–1139.

[113] Gregor Schoner and JA Kelso. "Dynamic pattern generation in behavioral and neural systems". In: *Science* 239.4847 (1988), pp. 1513–1520.

[114] Aneesha K Suresh et al. "Neural population dynamics in motor cortex are different for reach and grasp". In: *ELife* 9 (2020), e58848.

[115] Hannah R Sheahan, David W Franklin, and Daniel M Wolpert. "Motor planning, not execution, separates motor memories". In: *Neuron* 92.4 (2016), pp. 773–779.

[116] Carsen Stringer et al. "High-dimensional geometry of population responses in visual cortex". In: *Nature* 571.7765 (2019), pp. 361–365.

[117] Niru Maheswaranathan et al. "Universality and individuality in neural dynamics across large populations of recurrent networks". In: *Advances in neural information processing systems* 2019 (2019), p. 15629.

[118] Tim Van Gelder. "The dynamical hypothesis in cognitive science". In: *Behavioral and brain sciences* 21.5 (1998), pp. 615–628.

[119] JA Scott Kelso. "Multistability and metastability: understanding dynamic coordination in the brain". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 367.1591 (2012), pp. 906–918.

[120] William Bechtel. "Representing time of day in circadian clocks". In: *Knowledge and representation. Palo Alto, CA: CSLI Publications* (2011).

[121] Eberhard E Fetz. "Are movement parameters recognizably coded in the activity of single neurons?" In: *Behavioral and brain sciences* (1992), p. 154.

[122] Stephen H Scott. "Inconvenient truths about neural processing in primary motor cortex". In: *The Journal of physiology* 586.5 (2008), pp. 1217–1224.

[123] John P Cunningham and Zoubin Ghahramani. "Linear dimensionality reduction: Survey, insights, and generalizations". In: *The Journal of Machine Learning Research* 16.1 (2015), pp. 2859–2900.

[124] Sean R Bittner and John P Cunningham. "Approximating exponential family models (not single distributions) with a two-network architecture". In: *ICML Workshop on Invertible Neural Networks and Normalizing Flows* (2019).

[125] Peter Dayan et al. "The helmholtz machine". In: *Neural computation* 7.5 (1995), pp. 889–904.

[126] D. J. C. MacKay and M. N. Gibbs. "Density Networks". In: *Statistics and Neural Networks*. Oxford, 1997, pp. 129–146.

[127] Benigno Uria, Iain Murray, and Hugo Larochelle. "RNADE: The real-valued neural autoregressive density-estimator". In: *NIPS*. 2013, pp. 2175–2183.

[128] Oren Rippel and Ryan Prescott Adams. "High-dimensional probability estimation with deep density models". In: *arXiv preprint arXiv:1302.5125* (2013).

[129] Ian Goodfellow et al. "Generative Adversarial Nets". In: *NIPS 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2672–2680.

[130] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *arXiv* (Dec. 2013). eprint: `1312.6114`.

[131] Michalis Titsias and Miguel Lázaro-Gredilla. "Doubly stochastic variational Bayes for non-conjugate inference". In: *International Conference on Machine Learning*. 2014, pp. 1971–1979.

[132] Andrew Gelman et al. *Bayesian data analysis*. Vol. 2. CRC press Boca Raton, FL, 2014.

[133] Joshua B Tenenbaum, Thomas L Griffiths, and Charles Kemp. "Theory-based Bayesian models of inductive learning and reasoning". In: *Trends in cognitive sciences* 10.7 (2006), pp. 309–318.

[134] Peter McCullagh. "What Is a Statistical Model?" In: *The Annals of Statistics* 30.5 (2002), pp. 1225–1267.

[135] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York, 2001.

[136] Wenda Zhou et al. "Compressibility and Generalization in Large-Scale Deep Learning". In: *arXiv preprint arXiv:1804.05862* (2018).

[137] Martin J Wainwright, Michael I Jordan, et al. "Graphical models, exponential families, and variational inference". In: *Foundations and Trends® in Machine Learning* 1.1–2 (2008), pp. 1–305.

[138] Samuel Gershman and Noah Goodman. "Amortized inference in probabilistic reasoning". In: *Proceedings of the Annual Meeting of the Cognitive Science Society*. Vol. 36. 36. 2014.

[139] Andreas Stuhlmüller, Jacob Taylor, and Noah Goodman. "Learning stochastic inverses". In: *NIPS*. 2013, pp. 3048–3056.

[140] Christian Robert. *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer Science & Business Media, 2007.

[141] David JC MacKay and Linda C Bauman Peto. "A hierarchical Dirichlet language model". In: *Natural language engineering* 1.3 (1995), pp. 289–308.

[142] Yee Whye Teh et al. "Hierarchical Dirichlet Processes". In: *Journal of the American Statistical Association* 101.476 (2006), pp. 1566–1581. eprint: `https://doi.org/10.1198/016214506000000302`.

[143] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[144] Jonathan K Pritchard, Matthew Stephens, and Peter Donnelly. "Inference of population structure using multilocus genotype data". In: *Genetics* 155.2 (2000), pp. 945–959.

[145] Leemon Baird, David Smalenberger, and Shawn Ingkiriwang. "One-step neural network inversion with PDF learning and emulation". In: *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*. Vol. 2. IEEE. 2005, pp. 966–971.

[146] Esteban G Tabak, Eric Vanden-Eijnden, et al. "Density estimation by dual ascent of the log-likelihood". In: *Communications in Mathematical Sciences* 8.1 (2010), pp. 217–233.

[147] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP". In: *arXiv preprint arXiv:1605.08803* (2016).

[148] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. "i-RevNet: Deep Invertible Networks". In: *arXiv preprint arXiv:1802.07088* (2018).

[149] Marcin Andrychowicz et al. "Learning to learn by gradient descent by gradient descent". In: *NIPS*. 2016, pp. 3981–3989.

[150] George Papamakarios and Iain Murray. "Distilling intractable generative models". In: *Probabilistic Integration Workshop at Neural Information Processing Systems*. 2015.

[151] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. "Variational inference: A review for statisticians". In: *Journal of the American Statistical Association* 112.518 (2017), pp. 859–877.

[152] Lawrence K Saul and Michael I Jordan. “Exploiting tractable substructures in intractable networks”. In: *NIPS*. 1996, pp. 486–492.

[153] David Barber and Wim Wiegerinck. “Tractable variational structures for approximating graphical models”. In: *NIPS*. 1999, pp. 183–189.

[154] Matthew Hoffman and David Blei. “Stochastic structured variational inference”. In: *Artificial Intelligence and Statistics*. 2015, pp. 361–369.

[155] Dustin Tran, David Blei, and Edo M Airoldi. “Copula variational inference”. In: *NIPS*. 2015, pp. 3564–3572.

[156] Mariusz Bojarski et al. “Structured adaptive and random spinners for fast machine learning computations”. In: *arXiv preprint arXiv:1610.06209* (2016).

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

[158] Arthur Gretton et al. “A kernel two-sample test”. In: *Journal of Machine Learning Research* 13.Mar (2012), pp. 723–773.

[159] John P Cunningham, Krishna V Shenoy, and Maneesh Sahani. “Fast Gaussian process methods for point process intensity estimation”. In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 192–199.

[160] John P Cunningham et al. “Inferring neural firing rates from spike trains using Gaussian processes”. In: *NIPS*. 2008, pp. 329–336.

[161] Ryan Prescott Adams, Iain Murray, and David JC MacKay. “Tractable nonparametric Bayesian inference in Poisson processes with Gaussian process intensities”. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM. 2009, pp. 9–16.

[162] Matthew A Smith and Adam Kohn. “Spatial and temporal scales of neuronal correlation in primary visual cortex”. In: *Journal of Neuroscience* 28.48 (2008), pp. 12591–12603.

[163] Sean R Bittner et al. “Interrogating theoretical models of neural computation with emergent property inference”. In: *bioRxiv* (2021), p. 837567.

[164] Ryan N Gutenkunst et al. “Universally sloppy parameter sensitivities in systems biology models”. In: *PLoS Comput Biol* 3.10 (2007), e189.

[165] Kamil Erguler and Michael PH Stumpf. “Practical limits for reverse engineering of dynamical systems: a statistical analysis of sensitivity and parameter inferability in systems biology models”. In: *Molecular BioSystems* 7.5 (2011), pp. 1593–1602.

[166] Brian K Mannakee et al. “Sloppiness and the geometry of parameter space”. In: *Uncertainty in Biology*. Springer, 2016, pp. 271–299.

[167] John J Hopfield. “Neural networks and physical systems with emergent collective computational abilities”. In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[168] Haim Sompolinsky, Andrea Crisanti, and Hans-Jurgen Sommers. “Chaos in random neural networks”. In: *Physical review letters* 61.3 (1988), p. 259.

[169] Andrey V Olypher and Ronald L Calabrese. “Using constraints on neuronal activity to reveal compensatory changes in neuronal parameters”. In: *Journal of Neurophysiology* 98.6 (2007), pp. 3749–3758.

[170] Misha V Tsodyks et al. “Paradoxical effects of external modulation of inhibitory interneurons”. In: *Journal of neuroscience* 17.11 (1997), pp. 4382–4388.

[171] Kong-Fatt Wong and Xiao-Jing Wang. “A recurrent network mechanism of time integration in perceptual decisions”. In: *Journal of Neuroscience* 26.4 (2006), pp. 1314–1328.

[172] WR Foster, LH Ungar, and JS Schwaber. “Significance of conductances in Hodgkin-Huxley models”. In: *Journal of neurophysiology* 70.6 (1993), pp. 2502–2518.

[173] Astrid A Prinz, Dirk Bucher, and Eve Marder. “Similar network activity from disparate circuit parameters”. In: *Nature neuroscience* 7.12 (2004), pp. 1345–1352.

[174] Pablo Achard and Erik De Schutter. “Complex parameter landscape for a complex neuron model”. In: *PLoS computational biology* 2.7 (2006), e94.

[175] Dimitry Fisher et al. “A modeling framework for deriving the structural and functional architecture of a short-term memory microcircuit”. In: *Neuron* 79.5 (2013), pp. 987–1000.

[176] Timothy O’Leary et al. “Cell types, network homeostasis, and pathological compensation from a biologically plausible ion channel expression model”. In: *Neuron* 82.4 (2014), pp. 809–821.

[177] Leandro M Alonso and Eve Marder. “Visualization of currents in neural models with similar behavior and different conductance densities”. In: *Elife* 8 (2019), e42722.

[178] Cristopher M Niell and Michael P Stryker. “Modulation of visual responses by behavioral state in mouse visual cortex”. In: *Neuron* 65.4 (2010), pp. 472–479.

[179] Aman B Saleem et al. “Integration of visual motion and locomotion in mouse visual cortex”. In: *Nature neuroscience* 16.12 (2013), pp. 1864–1869.

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

[181] Paul Marjoram et al. “Markov chain Monte Carlo without likelihoods”. In: *Proceedings of the National Academy of Sciences* 100.26 (2003), pp. 15324–15328.

[182] Andreas Raue et al. “Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood”. In: *Bioinformatics* 25.15 (2009), pp. 1923–1929.

[183] Johan Karlsson, Milena Anguelova, and Mats Jirstrand. “An efficient method for structural identifiability analysis of large dynamic systems”. In: *IFAC Proceedings Volumes* 45.16 (2012), pp. 941–946.

[184] Keegan E Hines, Thomas R Middendorf, and Richard W Aldrich. “Determination of parameter identifiability in nonlinear biophysical models: A Bayesian approach”. In: *Journal of General Physiology* 143.3 (2014), pp. 401–416.

[185] Dhruva V Raman, James Anderson, and Antonis Papachristodoulou. “Delineating parameter unidentifiabilities in complex models”. In: *Physical Review E* 95.3 (2017), p. 032314.

[186] Gamaleldin F Elsayed and John P Cunningham. “Structure in neural population recordings: an expected byproduct of simpler phenomena?” In: *Nature neuroscience* 20.9 (2017), p. 1310.

[187] Cristina Savin and Gašper Tkačik. “Maximum entropy models as a tool for building precise neural controls”. In: *Current opinion in neurobiology* 46 (2017), pp. 120–126.

[188] Wiktor Młynarski et al. “Statistical analysis and optimality of neural systems”. In: *bioRxiv* (2020), p. 848374.

[189] Dustin Tran, Rajesh Ranganath, and David Blei. “Hierarchical implicit models and likelihood-free variational inference”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5523–5533.

[190] Pedro J Gonçalves et al. “Training deep neural density estimators to identify mechanistic models of neural dynamics”. In: *bioRxiv* (2019), p. 838383.

[191] Gabriel Loaiza-Ganem, Yuanjun Gao, and John P Cunningham. “Maximum entropy flow networks”. In: *International Conference on Learning Representations* (2017).

[192] Durk P Kingma and Prafulla Dhariwal. “Glow: Generative flow with invertible 1x1 convolutions”. In: *Advances in neural information processing systems*. 2018, pp. 10215–10224.

[193] Mark S Goldman et al. “Global structure, robustness, and modulation of neuronal models”. In: *Journal of Neuroscience* 21.14 (2001), pp. 5229–5238.

[194] Brendan K Murphy and Kenneth D Miller. “Balanced amplification: a new mechanism of selective amplification of neural activity patterns”. In: *Neuron* 61.4 (2009), pp. 635–648.

[195] Guillaume Hennequin, Tim P Vogels, and Wulfram Gerstner. “Optimal control of transient dynamics in balanced networks supports generation of complex movements”. In: *Neuron* 82.6 (2014), pp. 1394–1406.

[196] Giulio Bondanelli et al. “Population coding and network dynamics during OFF responses in auditory cortex”. In: *BioRxiv* (2019), p. 810655.

[197] Ashok Litwin-Kumar, Robert Rosenbaum, and Brent Doiron. “Inhibitory stabilization and visual coding in cortical circuits with multiple interneuron subtypes”. In: *Journal of neurophysiology* 115.3 (2016), pp. 1399–1409.

[198] Agostina Palmigiano et al. “Structure and variability of optogenetic responses identify the operating regime of cortex”. In: *bioRxiv* (2020).

[199] Chunyu A Duan et al. “Collicular circuits for flexible sensorimotor routing”. In: *Nature Neuroscience* (2021), pp. 1–11.

[200] Mark S Goldman. “Memory without feedback in a neural network”. In: *Neuron* 61.4 (2009), pp. 621–634.

[201] Giulio Bondanelli and Srdjan Ostojic. “Coding with transient trajectories in recurrent neural networks”. In: *PLoS computational biology* 16.2 (2020), e1007655.

[202] Scott A Sisson, Yanan Fan, and Mark Beaumont. *Handbook of approximate Bayesian computation*. CRC Press, 2018.

[203] Eve Marder and Allen I Selverston. *Dynamic biological networks: the stomatogastric nervous system*. MIT press, 1992.

[204] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[205] Lawrence Saul and Michael Jordan. “A mean field learning algorithm for unsupervised neural networks”. In: *Learning in graphical models*. Springer, 1998, pp. 541–554.

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

[207] W Keith Hastings. “Monte Carlo sampling methods using Markov chains and their applications”. In: (1970).

[208] Mark Girolami and Ben Calderhead. “Riemann manifold langevin and hamiltonian monte carlo methods”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.2 (2011), pp. 123–214.

[209] Ben Calderhead and Mark Girolami. “Statistical analysis of nonlinear dynamical systems using differential geometric sampling methods”. In: *Interface focus* 1.6 (2011), pp. 821–835.

[210] Andrew Golightly and Darren J Wilkinson. “Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo”. In: *Interface focus* 1.6 (2011), pp. 807–820.

[211] Oksana A Chkrebtii et al. “Bayesian solution uncertainty quantification for differential equations”. In: *Bayesian Analysis* 11.4 (2016), pp. 1239–1267.

[212] Juliane Liepe et al. “A framework for parameter estimation and model selection from experimental data in systems biology using approximate Bayesian computation”. In: *Nature protocols* 9.2 (2014), pp. 439–456.

[213] Sean R Bittner et al. “Degenerate solution networks for theoretical neuroscience”. In: *Computational and Systems Neuroscience Meeting (COSYNE), Lisbon, Portugal* (2019).

[214] Sean R Bittner et al. “Examining models in theoretical neuroscience with degenerate solution networks”. In: *Bernstein Conference 2019, Berlin, Germany* (2019).

[215] Marcel Nonnenmacher et al. “Robust statistical inference for simulation-based models in neuroscience”. In: *Bernstein Conference 2018, Berlin, Germany*. 2018.

[216] Deistler Michael et al. “Statistical inference for analyzing sloppiness in neuroscience models”. In: *Bernstein Conference 2019, Berlin, Germany*. 2019.

[217] Jan-Matthis Lueckmann et al. “Flexible statistical inference for mechanistic models of neural dynamics”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 1289–1299.

[218] George Papamakarios, David Sterratt, and Iain Murray. “Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows”. In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 837–848.

[219] Joeri Hermans, Volodimir Begy, and Gilles Louppe. “Likelihood-free mcmc with amortized approximate ratio estimators”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 4239–4248.

[220] Ricky TQ Chen et al. “Neural ordinary differential equations”. In: *Advances in neural information processing systems*. 2018, pp. 6571–6583.

[221] Xuechen Li et al. “Scalable gradients for stochastic differential equations”. In: *arXiv preprint arXiv:2001.01328* (2020).

[222] Maria Pia Saccomani, Stefania Audoly, and Leontina D’Angiò. “Parameter identifiability of nonlinear systems: the role of initial conditions”. In: *Automatica* 39.4 (2003), pp. 619–632.

[223] Stefan Hengl et al. “Data-based identifiability analysis of non-linear dynamical models”. In: *Bioinformatics* 23.19 (2007), pp. 2612–2618.

[224] Emmanuel Klinger, Dennis Rickert, and Jan Hasenauer. “pyABC: distributed, likelihood-free inference”. In: *Bioinformatics* 34.20 (2018), pp. 3591–3593.

[225] David S Greenberg, Marcel Nonnenmacher, and Jakob H Macke. “Automatic Posterior Transformation for Likelihood-Free Inference”. In: *International Conference on Machine Learning* (2019).

[226] Hirofumi Ozeki et al. “Inhibitory stabilization of the cortical network underlies visual surround suppression”. In: *Neuron* 62.4 (2009), pp. 578–592.

[227] Daniel B Rubin, Stephen D Van Hooser, and Kenneth D Miller. “The stabilized supralinear network: a unifying circuit motif underlying multi-input integration in sensory cortex”. In: *Neuron* 85.2 (2015), pp. 402–417.

[228] Guillaume Hennequin et al. “The dynamical regime of sensory cortex: stable dynamics around a single stimulus-tuned attractor account for patterns of noise variability”. In: *Neuron* 98.4 (2018), pp. 846–860.

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

[230] Henry Markram et al. “Interneurons of the neocortical inhibitory system”. In: *Nature reviews neuroscience* 5.10 (2004), p. 793.

[231] Bernardo Rudy et al. “Three groups of interneurons account for nearly 100% of neocortical GABAergic neurons”. In: *Developmental neurobiology* 71.1 (2011), pp. 45–61.

[232] Robin Tremblay, Soohyun Lee, and Bernardo Rudy. “GABAergic Interneurons in the Neocortex: From Cellular Properties to Circuits”. In: *Neuron* 91.2 (2016), pp. 260–292.

[233] Carsten K Pfeffer et al. “Inhibition of inhibition in visual cortex: the logic of connections between molecularly distinct interneurons”. In: *Nature Neuroscience* 16.8 (2013), p. 1068.

[234] Daniel J Felleman and David C Van Essen. “Distributed hierarchical processing in the primate cerebral cortex.” In: *Cerebral cortex (New York, NY: 1991)* 1.1 (1991), pp. 1–47.

[235] C Gardiner. *Stochastic methods: A Handbook for the Natural and Social Sciences*. 2009.

[236] Chunyu A Duan, Jeffrey C Erlich, and Carlos D Brody. “Requirement of prefrontal and midbrain regions for rapid executive control of behavior in the rat”. In: *Neuron* 86.6 (2015), pp. 1491–1503.

[237] Daniel P Mossing et al. “Antagonistic inhibitory subnetworks control cooperation and competition across cortical space”. In: *bioRxiv* (2021).