Machine Learning As Tool And Theory For Computational Neuroscience

Ari S. Benjamin
University of Pennsylvania

Machine Learning As Tool And Theory For Computational Neuroscience

Abstract

Computational neuroscience is in the midst of constructing a new framework for understanding the brain based on the ideas and methods of machine learning. This is effort has been encouraged, in part, by recent advances in neural network models. It is also driven by a recognition of the complexity of neural computation and the challenges that this poses for neuroscience's methods. In this dissertation, I first work to describe these problems of complexity that have prompted a shift in focus. In particular, I develop machine learning tools for neurophysiology that help test whether tuning curves and other statistical models in fact capture the meaning of neural activity. Then, taking up a machine learning framework for understanding, I consider theories about how neural computation emerges from experience. Specifically, I develop hypotheses about the potential learning objectives of sensory plasticity, the potential learning algorithms in the brain, and finally the consequences for sensory representations of learning with such algorithms. These hypotheses pull from advances in several areas of machine learning, including optimization, representation learning, and deep learning theory. Each of these subfields has insights for neuroscience, offering up links for a chain of knowledge about how we learn and think. Together, this dissertation helps to further an understanding of the brain in the lens of machine learning.

Degree Type

Dissertation

Degree Name

Doctor of Philosophy (PhD)

Graduate Group

Bioengineering

First Advisor

Konrad P. Kording

Keywords

Computational neuroscience, Learning, Machine learning, Neural networks, Neurophysiology, Theoretical neuroscience

Subject Categories

Artificial Intelligence and Robotics | Neuroscience and Neurobiology

MACHINE LEARNING AS TOOL AND THEORY FOR COMPUTATIONAL NEUROSCIENCE

Ari Benjamin

A DISSERTATION

in

Bioengineering

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2022

ACKNOWLEDGEMENTS

One page cannot contain everyone who has shaped me and this dissertation. Thank you, everyone. A little bit of you is in each of these pages.

First, I want thank my advisor Konrad for taking a chance on a PhD student in a different field wanting to start it all over. My life took a different turn in that moment. Konrad gifts his lab a true freedom. In the following years I was free to think, to grow, and to discover neuroscience and find my place within it. Few scientists have such a willingness to question the big picture. I'm especially thankful for his conviction that science is a social enterprise, and everything that meant for our lab and broader scientific community. I feel very fortunate to have "grown up" in this kind, curious, and lively environment and to have absorbed some of his wise approach to science.

I'd like to thank all the members of the Kording lab, past and present, who showed me how to ask better questions and find meaning within computational neuroscience. It was easy to find purpose with the easy guidance of friends who knew their stuff.

Computational neuroscience is a wonderful community, and I'm lucky to have found myself within it. The COSYNE conference community, Neuromatch, Penn's computational neuroscience, and the CCN conference have made this science very enjoyable. Thank you for such an open and collaborative community.

I have many collaborators to thank. Thank you to Josh Glaser and Roozbeh Farhoodi, with whom it was always a pleasure to work aside. Pavan Ramkumar kindly shared his knowledge and showed me gracious patience when I was just a first-year. Thanks to David Rolnick, whose lively discussion I always looked forward to. The same is true for Richard Lange, who I also wish to thank for his thoughtful insight on many issues in representation learning and Bayesian inference. I have a warm thanks for Professor Lee Miller and Matthew Smith and their collaboration, and especially to Matt who has lent a kind perseverance as our paper has had its odyssey in peer review. Ling-Qi Zhang has been a partner in thought in wonderful ways. For showing me the world of visual psychophysics, I am also thankful to Cheng Qiu and Alan Stocker.

For shaping me as a scientist, I also must thank the incredible educators of Northampton, MA, as well as every physics professor at Williams College. You showed me how to think and demonstrated a truly infectious love of teaching. Also to Joe Cruz and Safa Zeki at Williams who, though they do not know it, sparked an interest in the mind and brain that grew to this dissertation.

Thank you, Kate, for encouraging me to stay true to what I love, for building a life with adventure, care, and curiosity, and for showing me what an active, ethical engagement with society can look like.

Finally, I am infinitely grateful to my parents, who saw the creative, curious scientific spark in me at a young age and did everything to encourage it. Their boundless care for me and my wellbeing made me everything I am.

ABSTRACT

MACHINE LEARNING AS TOOL AND THEORY FOR COMPUTATIONAL NEUROSCIENCE

Ari S. Benjamin

Konrad P. Kording

Computational neuroscience is in the midst of constructing a new framework for understanding the brain based on the ideas and methods of machine learning. This is effort has been encouraged, in part, by recent advances in neural network models. It is also driven by a recognition of the complexity of neural computation and the challenges that this poses for neuroscience's methods. In this dissertation, I first work to describe these problems of complexity that have prompted a shift in focus. In particular, I develop machine learning tools for neurophysiology that help test whether tuning curves and other statistical models in fact capture the meaning of neural activity. Then, taking up a machine learning framework for understanding, I consider theories about how neural computation emerges from experience. Specifically, I develop hypotheses about the potential learning objectives of sensory plasticity, the potential learning algorithms in the brain, and finally the consequences for sensory representations of learning with such algorithms. These hypotheses pull from advances in several areas of machine learning, including optimization, representation learning, and deep learning theory. Each of these subfields has insights for neuroscience, offering up links for a chain of knowledge about how we learn and think. Together, this dissertation helps to further an understanding of the brain in the lens of machine learning.

Table of Contents

ACKNOWLEDGEMENTS .... **II**

ABSTRACT .... **IV**

LIST OF ILLUSTRATIONS .... **VII**

CHAPTER 1: INTRODUCTION **1**

SECTION 1: NEUROPHYSIOLOGY PRACTICE AND THE COMPLEXITY OF SENSORY CORTEX 2
SECTION 2: LEARNING AND ITS CONSEQUENCES 6

STRUCTURE OF THIS DISSERTATION **12**

CHAPTER 2: HUE TUNING CURVES IN V4 CHANGE WITH VISUAL CONTEXT **13**

FOREWORD 13
ABSTRACT 14
INTRODUCTION 14
MATERIALS AND METHODS 16
RESULTS 28
DISCUSSION 41
REFERENCES 46
SUPPLEMENTARY INFORMATION 50

CHAPTER 3: MODERN MACHINE LEARNING AS BENCHMARK FOR ENCODING MODELS **55**

FOREWORD 55
ABSTRACT 56
INTRODUCTION 56
MATERIALS AND METHODS 58
RESULTS 66
DISCUSSION 74
REFERENCES 77

CHAPTER 4: (INTERLUDE) THE FOUR ROLES OF SUPERVISED MACHINE LEARNING IN SYSTEMS NEUROSCIENCE **81**

FOREWORD 81
ABSTRACT 82
INTRODUCTION 82
ROLE 1: SOLVING ENGINEERING PROBLEMS 85
Neural Activity / Function *86*
Neuroanatomy / Structure *88*
ROLE 2: IDENTIFYING PREDICTIVE VARIABLES 89
Neural Activity / Function *90*
Neuroanatomy / Structure *92*
ROLE 3: BENCHMARKING SIMPLE MODELS 93
Neural Activity / Function *95*
Neuroanatomy / Structure *95*
ROLE 4: SERVING AS A MODEL FOR THE BRAIN 95
DISCUSSION 101
REFERENCES 103

CHAPTER 5: LEARNING TO INFER IN RECURRENT BIOLOGICAL NETWORKS.. 111

FOREWORD.. 111
ABSTRACT.. 112
INTRODUCTION.. 112
BACKGROUND: THE GOAL OF LEARNING.. 114
RESULTS.. 117
DISCUSSION.. 126
METHODS.. 128
REFERENCES.. 132

CHAPTER 6: MEASURING AND REGULARIZING NETWORKS IN FUNCTION SPACE............................... 136

FOREWORD.. 136
ABSTRACT.. 137
INTRODUCTION.. 137
RESULTS.. 139
DISCUSSION.. 154
REFERENCES.. 156
APPENDIX.. 158

CHAPTER 7: EFFICIENT NEURAL CODES NATURALLY EMERGE THROUGH GRADIENT DESCENT LEARNING 165

FOREWORD.. 165
ABSTRACT.. 166
INTRODUCTION.. 166
RESULTS.. 169
DISCUSSION.. 178
METHODS.. 182
REFERENCES.. 191
SUPPLEMENTAL FIGURES.. 196
APPENDIX.. 198

CHAPTER 8: CONCLUSIONS.. 207

Chapters 2 and 3: neurophysiology.. 207
Chapter 5: Probabilistic representation learning.. 208
Chapter 6: Neural network optimization.. 209
Chapter 7: The sensory consequences of learning algorithms.. 210
Conclusion.. 211
References.. 211

APPENDIX: PUBLICATIONS.. 213

Appearing as chapters in this dissertation.. 213
Publications not mentioned in this dissertation.. 213

List of Illustrations

FIGURE 2-1: 29
FIGURE 2-2 30
FIGURE 2-3 31
FIGURE 2-4 34
FIGURE 2-5 38
FIGURE 2-6 39
FIGURE 2-7 50
FIGURE 2-8 51
FIGURE 2-9 52
FIGURE 2-10 54
FIGURE 3-1: 67
FIGURE 3-2 68
FIGURE 3-3 69
FIGURE 3-4 70
FIGURE 3-5 71
FIGURE 3-6 72
FIGURE 3-7 73
FIGURE 4-1 83
FIGURE 4-2 84
FIGURE 4-3 85
FIGURE 4-4 100
FIGURE 5-1 116
FIGURE 5-2 118
FIGURE 5-3 122
FIGURE 5-4 124
FIGURE 5-5 125
FIGURE 6-1 140
FIGURE 6-2 143
FIGURE 6-3 144
FIGURE 6-4 147
FIGURE 6-5 153
FIGURE 6-6 158
FIGURE 6-7 158
FIGURE 6-8 159
FIGURE 6-9 159
FIGURE 6-10 159
FIGURE 6-11 160
FIGURE 6-12 161
FIGURE 7-1 170
FIGURE 7-2 171
FIGURE 7-3 172
FIGURE 7-4 173
FIGURE 7-5 174

FIGURE 7-6 178
FIGURE 7-7 196
FIGURE 7-8 197

Chapter 1: Introduction

Neural network models are playsets for a young neuroscience. They are simpler arenas in which to grow and to theorize. They allow neuroscience to apply its paradigms and, with complete access and control, see what can be learned.

Neural networks embody a commitment to understanding the brain in the lens of its function. For this aim, it helps to have some knowledge of what it takes to engineer that function. For example, the organization of spider silk might be understood by how it supplies strength to keep its shape, lightness to reduce its weight, and toughness to resist a break, which are all concepts of mechanical engineering (Keten, Xu, Ihle, & Buehler, 2010). Similarly, the way in which the brain learns may be best understood with the concepts of the engineering of learning machines.

Machine learning provides neuroscience with a language and set of ideas for formalizing learning (Marblestone, Wayne, & Kording). This includes knowledge about the obstacles that make learning from experience difficult. These problems have been solved, somehow, by evolution. Finding nature's solutions to the problems of learning (as defined by machine learning) is a higher-level, more normative approach than is traditional in neuroscience. It does not begin from synaptic plasticity, for example. Instead, this approach describes what is necessary of learning in the abstract and what this means for hypotheses of learning in the brain.

A focus on neural networks also represents an embrace of complexity. In artificial intelligence, the desired programs (for example, successfully playing Go at super-human levels) are often too complex to be listed as a set of human-intelligible rules. The engineering solution is to instead design a training regimen for a neural network and let a computation emerge. Recently, some in computational neuroscience have argued to similarly shift the focus of research away from describing computations and instead focus on the processes by which they arise (Richards et al., 2019). These many be easier to discover and describe at an abstract level, assuming the right theoretical language is used.

This shift has its roots in an old controversy in neuroscience. How complex is neural computation, exactly, and what does this mean about how we, as neuroscientists, should understand it? In sensory processing and in particular the neurophysiology of early vision (long a model for neuroscience practice), neural computation has been argued to be both too complex for common methods (Bruno A Olshausen & Field, 2005) and within reach (Rust & Movshon, 2005). Time has not resolved the argument, although new methods as well as arguments from machine learning are now adding weight to complexity's side (Cadena et al., 2019; Lillicrap & Kording, 2019). This dissertation begins with two studies that add to this ongoing and field-wide debate. Specifically, these introduce tools that evaluate the success of techniques for understanding neural responses, paying particular attention to the neurophysiology of vision. These studies provide a grounding for a broader discussion about which methods for understanding neural computation are likely to be fruitful.

This dissertation is organized into two sections. **Section 1** discusses complexity within the practice of neurophysiology. These studies center the use of machine learning techniques as tools. **Section 2** deals with neural network models of learning and perception. This work is situated between artificial intelligence and neuroscience, and asks, broadly, how these disciplines can mutually benefit.

Section 1: Neurophysiology practice and the complexity of sensory cortex

What is the meaning of the activity of single neurons in the visual cortex? This question has been the subject of some of the most intense and dedicated study in all of neuroscience. In the tradition of Hubel and Wiesel, a predominant approach has been to present visual stimuli to animals while electrically recording the activity of neurons, and then to describe a model (conceptual or mathematical) that describes when neurons typically fire. This approach has been successful in revealing many unexpected aspects about what happens during vision, and it remains a mainstay of the perceptual neurosciences.

Arguably the central difficulty for single-neuron neurophysiology that one only records the responses to the select stimuli shown in the experiment. One cannot show all possible stimuli. From these concrete instances, the challenge is to extract an understanding that is more general. This perennial problem for the sciences (and epistemology, no less) carries precise meaning for neurophysiology. Given some observations, one should say something about the response to different stimuli than those shown. Models and descriptions necessarily generalize from limited instances.

Most critiques of typical methods for neurophysiology are critiques of an *overgeneralization* of experimental results that create a misleading or false picture of neural function. In their "neurophysiology" of a microprocessor, for example, Jonas & Kording illustrate such overgeneralization in practice (Jonas & Kording, 2017). After observing that the voltage of a transistor correlates with an action in a video game, one could overgeneralize to say that the transistor's "role" is to encode that action. This is an extrapolation and, in this case, an incorrect one. Olshausen & Field have also highlighted problems of overgeneralization in the study of V1 responses (Bruno A Olshausen & Field, 2005, 2006). They argue that experiments are systematically biased towards certain inputs (simple and artificial stimuli) as well as certain neurons (those with high firing rates on those stimuli), which results in failures when generalizing to other contexts. In light of these critiques, the onus lies with neurophysiology to establish exactly how far our models generalize, or in other words, to prove that an experiment does in fact establish the meaning of neural activity.

Statisticians and epistemologists alike recognize that generalizing from limited experience requires assumptions. What are the assumptions of typical approaches in neurophysiology? How might these assumptions be empirically verified? The concrete aim of Section 1 is to introduce tools for verification for neurophysiologists that quantify how much a model will generalize. The two sets of methods that I will focus on are *tuning curves* and more broadly *encoding models*.

Chapter 2: Tuning curves

A tuning curve characterizes whether or not a neuron is "tuned for" a variable describing the sensory world, like a tuning fork might be tuned for a particular acoustic frequency. Tuning curves clearly have meaning with regards to the stimuli and responses from which they were constructed, but what meaning do they have about how the brain processes other stimuli? Often, the interpretation is quite general. If a neuron responds strongly to certain orientations of a bar of light, it might be said to encode whether that orientation is present in its receptive field (in general). If this interpretation is taken, it implies a strong assumption about the meaning of neural activity in other contexts.

In Chapter 2, I present a method that allows testing the assumption that tuning curves generalize beyond the laboratory context. Specifically, this is a method to estimate tuning from responses in more complicated contexts, and in particular naturalistic stimuli. This technique was demonstrated in a collaboration with Prof. Matthew Smith of Carnegie Mellon University for tuning to hue in macaque area V4. By testing for a change of tuning with context, one can assess whether this description of activity is a correct generalization from responses to laboratory stimuli.

Chapter 3: Encoding models

Tuning curves are just one type of *encoding model*, statistical models for describing a neural response in terms of the stimulus. An encoding model takes as its central metaphor the idea of a "neural code", that activity is like a cipher for sensory information (Aljadeff, Lansdell, Fairhall, & Kleinfeld, 2016; Rieke, Warland, Van Steveninck, & Bialek, 1999). Whereas tuning curves are built by directly tabulating the stimulus/response function, encoding models may also be obtained by fitting regression models. Different assumptions can be made depending on the form of the regression. Most often neurophysiologists select models by considering ease of use, interpretability, and the quantitative success of fitting neural activity.

In Chapter 3, I describe a method that can verify that the assumptions embedded in an encoding model are well-suited for the neurons in question. This approach is simple in principle: apply the methods and techniques of black-box machine learning to predict neural activity, and then take this predictive ability as a performance benchmark for simpler, hypothesis-driven statistical models. This approach circumvents some of the drawbacks of other standard verification methods, such as repeating a stimulus multiple times to establish the noise level. If performance is near the benchmark, one can be more confidence that a statistical description of neural activity is an accurate one.

Summary

Tuning curves and encoding models make assumptions about neural activity in other contexts. These assumptions are necessary but must be verified in order to be trusted. Each project in Section 1 introduces a tool for such a verification as well as demonstrations on recordings of single neurons in macaque cortex.

Such verifications provide an opportunity for an honest evaluation of encoding models in future experiments, and also inform a broader discussion about encoding models and their interpretation. In general, given a new tuning curve or encoding model of cortical activity, how much should a neurophysiologist that summarizes a general computation? This is a generalization of past experiments to future, unperformed experiments. Since even generalizations about generalizations require assumptions, neuroscience must consider arguments for the likely form and complexity of sensory processing.

Outlook: how might we understand an artificial neural network?

Neurophysiology ought to be easy in artificial neural networks (ANNs). There are no unobserved confounds like attention or neuromodulation. All nodes are perfectly visible and there are no hidden variables. How well might a neurophysiologist succeed at parsing the meaning of units in these networks?

This is not purely hypothetical; many in artificial intelligence have attempted such a thing. Scientists have ported over methods such as tuning curves to see what might be illuminated (Goh et al., 2021). New methods have also been developed, such as visualizing the stimuli that maximally excite a specific unit or layer (Olah, Mordvintsev, & Schubert, 2017). Yet, while these methods give an intuitive understand of how ANNs work, it has become clear that one cannot use them to predict how an ANN will classify and respond to new inputs.

It is a concerning possibility that the extraordinary complexity of computation in large artificial networks may evade a complete understanding. If one measures complexity by how much information would be required to describe how a system works, any description would need to be extraordinarily lengthy and, perhaps, too lengthy for a human to internalize (Lillicrap & Kording, 2019). New concepts and theories might help break down and package these computations, but at the moment it is clear that we do not currently have the tools to understand how ANNs 'see' their inputs and come to their decisions.

Section 2: Learning and its consequences

Complex systems can emerge from simple rules. Recently, some in computational neuroscience have argued to shift the focus away from the computations performed in the brain and towards how they are learned (Lillicrap & Kording, 2019; Richards et al., 2019). Proponents of this shift often argue that in ANNs, the emergent computation is much more difficult to understand than the factors set by the practitioner – the architecture, the learning objective, and the learning algorithm. The same may be true of brains. Rather than deconstructing the computation of networks, this effort seeks to understand how they learn and to what ends. It is a 'deep learning framework' for neuroscience.

A deep learning framework is not an effort to map the components of modern deep learning systems onto the brain. Rather, it means adopting a computational language developed by machine learning and statistical learning theory – disciplines that are concerned with describing learning and learnability in the abstract. This rich language has enormous potential to help understand how brains learn and how this shapes the meaning of neural activity.

In Section 2 of this dissertation, I embrace this approach to understanding the brain in the course of three chapters. These projects each use deep learning models and theory in order to reason about learning in the brain. Each takes up one of the following questions: what are potential *learning objectives* for sensory cortex? What are potential *learning algorithms*? And what are the *sensory consequences* for having learned with such algorithms?

A learning objective: representation learning

One of the core concepts in a learning framework is the objective or goal of learning. This is a teleological interpretation of neuronal plasticity. Much of this dissertation focuses on perception. What is the objective of plasticity in sensory cortex?

In Chapter 5, I examine the hypothesis that the sensory cortex aims to form internal representations of the external world. This objective of *representation learning* is central concept in neuroscience (though not without critique (Baker, Lansdell, & Kording, 2021; Brette, 2019)). As commonly defined, representations are transformations of sensory data into forms that are more useful to an organism. Good representations might highlight the true organizing principles of the world (Kersten, Mamassian, & Yuille, 2004) or encode information as best as possible while using minimal energy (Barlow, 1961). In this broad frame, the goal of sensory learning is to form useful representations.

Representations are also key concept in machine learning. There, many works have attempted to formalize how to produce useful representations. One popular approach imagines that good sensory systems first create a *model* of the sensory world, and then *infer* the representations in that model that explain sensory data (Yuille & Kersten, 2006). Perception might be like a video-game rendering engine with representations of the lighting, materials, and objects that best explain a visual scene; a 'simulation in the mind' (Ullman, Spelke, Battaglia, & Tenenbaum, 2017). Many different types of models are possible, making this a general and flexible way to describe representations.

What would be required if the brain learns representations in this way? In this Chapter, I aim to characterize the specific problems that this introduces for neural circuits. I then develop theories about possible solutions that may be taken by sensory cortex. These theories draw from machine learning techniques for representation learning, but are adapted so as to serve as biological hypotheses. The goal is to evaluate whether this class of objectives is a candidate for a description of the objective of sensory learning.

A learning algorithm: improving upon gradient descent

A *learning algorithm* is a set of rules prescribing how a system ought to change over time. A deep learning framework for neuroscience aims to build a computational understanding of the brain's learning algorithms, abstracted from the level of synaptic plasticity and its cellular implementation. Because this effort is in its infancy, a crucial first step is to identify and understand the candidate algorithms.

Chapter 6 represents a foray into the fields of deep learning optimization and deep learning theory. These fields have identified algorithms that work well on deep learning systems, and then sought to understand why they work. The baseline algorithm is gradient descent, in which all parameters change proportional to their effect upon the output. Yet in deep learning, almost all papers now use different algorithms that, in practice, produce better networks with less training data. This chapter discusses a theoretical perspective on algorithms that improve upon gradient descent, and then uses this to derive a new algorithm for learning.

Of all that machine learning has to offer a neuroscience of learning, perhaps the most important is simply a recognition of how difficult and unlikely learning really is. Like scientists pulling general knowledge from limited experiments, learning systems must pull generalities from limited experience. This requires making assumptions (David H. Wolpert & Macready, 1997). In deep learning systems, the assumptions come as much from the learning algorithm as the constraints of the system itself (Zhang, Bengio, Hardt, Recht, & Vinyals, 2021). Somehow, on certain problems, learning algorithms 'choose' to learn knowledge that generalizes well. The network, algorithm, and data conspire together for effective learning. Since the brain, too, has a staggering amount of plastic parameters, the theories that describe why ANNs generalize are likely to be useful for describing why we learn so effectively, too.

The sensory consequences of learning algorithms

A deep learning framework does not require abandoning a study of responses for a study of objectives and algorithms. Understanding learning may be the best ways to gain insight about responses, as well. This requires that bridges be built between neuroscience and deep learning theory.

An important lesson of deep learning is that the learning algorithm leaves permanent traces upon the network and the function it implements. The algorithm is not an incidental and ultimately ignorable process. The traces left by are learning are in fact so important that useful learning in large neural networks is not possible without them (Zhang et al., 2021). These traces are the bias of the learning algorithm, which are effectively assumptions about what is important to learn. As mentioned in the previous section, such assumptions are necessary for learning. Neural network representations thus must carry the imprint of algorithms that generalize well from limited data.

This perspective is new to neuroscience. Usually, when explaining why neural representations take one form or another, neuroscience reaches to normative explanations and describes how the *adult* representation is optimal in some sense. A hypothesis of this flavor is that sensory representations minimize the number of spikes required to encode external information because of evolutionary pressure on energy usage (Bruno A. Olshausen & Field, 1996; Rao & Ballard, 1999). Though pressures like this certainly exist, there also exists an evolutionary pressure to learn effectively as an infant. Explaining adult representations via their emergence from effective learning algorithms introduces a new type of explanation, this time referencing machine learning theory.

In Chapter 7, I show how such ideas can explain a set of findings in psychophysics and neurophysiology about sensory representations. In humans and other animals, sensory systems tend to better encode the features of the world that are more common, especially for low-level features like orientation and color (Wei & Stocker, 2017). In information theory, such a strategy represents an *efficient code* – a code that makes best use of a channel with limited capacity. In a collaboration with the lab of Alan Stocker, we use artificial neural networks as model systems to demonstrate that gradient descent learning naturally results in efficient codes, as well.

Learning in the brain is unlikely to be gradient descent, precisely, but a similar principle is likely to be in play. As the algorithms of sensory learning come into focus for neuroscience, it will be interesting to describe how these shape what our brains choose to learn from experience.

Summary and outlook

To understand neural computation, a learning framework looks to its source. Yet, like computation, learning can be understood at many levels. A deep learning or machine learning framework for neuroscience aims for a high-level, normative understanding in terms able to be abstracted from the brain's implementation.

The projects in this dissertation endeavor to use modern theories of machine learning to advance neuroscience. Along the way they engage with a number of leading theories of sensory processing, asking, always, how might these be learned? To find answers, I pull ideas from multiple subfields of artificial intelligence, from optimization to generative adversarial networks to deep learning theory. These machine learning perspectives offer new types of explanations and predictions for neuroscience.

References

Aljadeff, J., Lansdell, B. J., Fairhall, A. L., & Kleinfeld, D. (2016). Analysis of neuronal spike trains, deconstructed. *Neuron, 91*(2), 221-259.

Baker, B., Lansdell, B., & Kording, K. (2021). A philosophical understanding of representation for neuroscience. *arXiv preprint arXiv:2102.06592*.

Barlow, H. B. (1961). Possible principles underlying the transformation of sensory messages. *Sensory communication, 1*(01).

Brette, R. (2019). Is coding a relevant metaphor for the brain? *Behavioral and brain sciences, 42*.

Cadena, S. A., Denfield, G. H., Walker, E. Y., Gatys, L. A., Tolias, A. S., Bethge, M., & Ecker, A. S. (2019). Deep convolutional models improve predictions of macaque V1 responses to natural images. *PLoS Computational Biology, 15*(4), e1006897.

Goh, G., Cammarata, N., Voss, C., Carter, S., Petrov, M., Schubert, L., . . . Olah, C. (2021). Multimodal neurons in artificial neural networks. *Distill, 6*(3), e30.

Jonas, E., & Kording, K. P. (2017). Could a neuroscientist understand a microprocessor? *PLoS Computational Biology, 13*(1), e1005268.

Kersten, D., Mamassian, P., & Yuille, A. (2004). Object perception as Bayesian inference. *Annu. Rev. Psychol., 55*, 271-304.

Keten, S., Xu, Z., Ihle, B., & Buehler, M. J. (2010). Nanoconfinement controls stiffness, strength and mechanical toughness of β-sheet crystals in silk. *Nature materials, 9*(4), 359-367.

Lillicrap, T. P., & Kording, K. P. (2019). What does it mean to understand a neural network? *arXiv preprint arXiv:1907.06374*.

Marblestone, A. H., Wayne, G., & Kording, K. P. (2016). Toward an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience*, 94.

Olah, C., Mordvintsev, A., & Schubert, L. (2017). Feature visualization. *Distill, 2*(11), e7.

Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature, 381*(6583), 607-609.

Olshausen, B. A., & Field, D. J. (2005). How close are we to understanding V1? *Neural computation, 17*(8), 1665-1699.

Olshausen, B. A., & Field, D. J. (2006). What is the other 85 percent of V1 doing. *L. van Hemmen, & T. Sejnowski (Eds.), 23*, 182-211.

Rao, R. P. N., & Ballard, D. H. (1999). Hierarchical Predictive Coding Model Hierarchical Predictive Coding of Natural Images. *Nature Neuroscience, 2*(1), 79-87. Richards, B. A., Lillicrap, T. P., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., . . . Ganguli, S. (2019). A deep learning framework for neuroscience. *Nature Neuroscience, 22*(11), 1761-1770.

Rieke, F., Warland, D., Van Steveninck, R. d. R., & Bialek, W. (1999). *Spikes: exploring the neural code*: MIT press.

Rust, N. C., & Movshon, J. A. (2005). In praise of artifice. *Nature Neuroscience, 8*(12), 1647-1650.

Ullman, T. D., Spelke, E., Battaglia, P., & Tenenbaum, J. B. (2017). Mind games: Game engines as an architecture for intuitive physics. *Trends in cognitive sciences, 21*(9), 649-665.

Wei, X. X., & Stocker, A. A. (2017). Lawful relation between perceptual bias and discriminability. *Proceedings of the National Academy of Sciences of the United States of America, 114*(38), 10244-10249. doi:10.1073/pnas.1619153114

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation, 1*(1), 67-82.

Yuille, A., & Kersten, D. (2006). Vision as Bayesian inference: analysis by synthesis? *Trends in cognitive sciences, 10*(7), 301-308. doi:10.1016/j.tics.2006.05.002

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM, 64*(3), 107-115.

Structure of this dissertation

This dissertation is unusual in its scope. Its chapters span multiple subfields of neuroscience and artificial intelligence. To unite these works into one dissertation, each chapter begins with a brief Foreword that provides a more general contextualization.

Machine learning plays several distinct roles in this dissertation. In a review published in *Progress in Neurobiology*[1], my co-authors and I described four categories of ways in which machine learning can assist neuroscience. These four roles are described in (Interlude) The Four Roles of Supervised Machine Learning in Systems Neuroscience:

- **Role 1**: to help create solutions to engineering problems.
- **Role 2:** to help in identifying variables that are predictive of something, like neural activity or disease.
- **Role 3:** to set benchmarks for simple models of the brain.
- **Role 4:** to itself serve as a model for understanding the brain.

In the Foreword to each chapter, I mention which of these four categories that chapter best represents.

Each chapter represents work that has been published in a peer-reviewed journal or is available online as a preprint. The list of publications can be found following the conclusion.

[1] Glaser, Joshua I.*, Ari S. Benjamin*, Roozbeh Farhoodi*, and Konrad P. Kording. "The roles of supervised machine learning in systems neuroscience". *Progress in neurobiology*. 2019 Apr 1;175:126-37.
* denotes co-first authorship

Chapter 2: Hue tuning curves in V4 change with visual context

Foreword

A common method of investigating visual cortex is to characterize whether neurons are 'tuned' to visual features. How much does this approach tell us a neuron's general role in vision?

Tuning curves allow generalizing from stimuli by making assumptions about responses to unseen stimuli. One key assumption is that the encoded parameter (e.g. orientation) affects activity in a similar way on many stimuli. One way to verify such a claim is to measure tuning in other contexts. If tuning describes a general role in processing, it should generalize to stimuli not presented.

This chapter provides a method to establish how far a tuning curve might generalize. It is an example of **Role 1:** machine learning as an engineering tool for neuroscience.

One way to confirm this assumption is to measure tuning in other contexts. In vision, the context of natural scenes is the ethologically relevant one, but it is difficult to measure tuning on natural scenes because each image is different in many ways (not just the tuning curve's feature). The nonlinearity of the cortical response also complicates this effort. Here, we developed a method to estimate tuning despite these difficulties. This can help to establish how far a tuning curve might generalize.

This chapter is reproduced from a paper available online as a preprint[2] and is currently in peer review. It was been presented as a poster at the Bernstein Conference for Computational Neuroscience in 2019 and the Conference of Cognitive Computational Neuroscience in 2017.

[2] Benjamin, Ari S., Pavan Ramkumar, Hugo Fernandes, Matthew A. Smith, and Konrad Paul Kording. "Hue tuning curves in V4 change with visual context." bioRxiv (2020): 780478.

Abstract

Neurons are often characterized by tuning curves estimated from responses to a set of artificial stimuli. A critical question for any tuning curve is how much tuning measured with one stimulus set reveals about tuning to a new set. Here we ask this question for neurons in macaque V4 by estimating tuning to hue from a set of natural scene stimuli and again from a set of simple color stimuli. We found that hue tuning was strong in each dataset but was not correlated across the datasets. This is expected if neurons have strong mixed selectivity. We also show how such mixed selectivity may be useful for transmitting information about multiple dimensions of the world. Our findings emphasize the importance of confirming empirically that tuning curves in higher sensory areas generalize to naturalistic stimuli.

Introduction

Neurophysiology has long investigated the visual cortex by asking how its various areas encode the visual world. In order to simplify experimental design, the majority of early work constructed stimuli sets in which only a few key visual parameters varied. By then observing which areas show corresponding changes in neural activity, the visual cortex can be described in terms of the variables for which neurons are highly tuned. This program has been successful in characterizing how the response properties of neurons in the ventral stream ascend in complexity. V1 is discussed as responding to "edge-detecting" Gabor filters (Carandini et al., 2005), V2 to variations in local curvature (Hegdé & Van Essen, 2003), V4 to more complex shapes (Pasupathy & Connor, 2001), and IT to specific objects and faces (Hung, Kreiman, Poggio, & DiCarlo, 2005), which together have inspired the theory that object recognition proceeds via hierarchical image representations (Connor, Brincat, & Pasupathy, 2007; Logothetis & Sheinberg, 1996).

In recent decades there has been a greater interest in understanding how much the response to simple, parameterized stimuli is informative about how the cortex encodes naturalistic stimuli. In addition to representing a field-wide shift towards ethology and natural paradigms, this question pertains to how much tuning curves estimated from artificial stimuli are good models of neurons' general functional role in visual processing. If tuning varies widely and unpredictably with context, the knowledge gained from simplified stimuli would be unique and particular to the tested stimuli alone. It is therefore critical to test how much tuning inferred from simplified stimuli is informative of tuning for more complicated stimuli.

In principle, such a question could be asked by steadily increasing the number of stimulus parameters that are varied until the complexity of stimuli approaches that of natural scenes. The original tuning curve could then be appreciated in the context of all others, and all interactions with other variables characterized. However, this approach is prohibitive because the number of stimuli required to be displayed scales exponentially with the number of parameters varied. Any practical experiment of this type would need to leave many potential parameters unvaried.

An alternative paradigm is to characterize neurons directly from their responses to natural images by regressing models of the visual response (David & Gallant, 2005; David, Vinje, & Gallant, 2004; Felsen & Dan, 2005; Simoncelli & Olshausen, 2001; Touryan, Felsen, & Dan, 2005; Vinje & Gallant, 2000). This is made possible by constraining what visual encodings are possible. For early areas, scientists often fit rather simple encoding models such as linear or linear-nonlinear models. This has revealed that some aspects of the V1 response are different for natural images (David et al., 2004), which limits the utility of characterizing artificial stimuli responses (David & Gallant, 2005; Bruno A Olshausen & Field, 2006). For higher areas, however, a regression methodology has not yet allowed such a comparison. In this work we focus on area V4. While the response of V4 to natural scenes been studied (Cowley, Williamson, Clemens, Smith, & Byron, 2017; Gallant, Connor, & Van Essen, 1998; Mazer & Gallant, 2003; T. O. Sharpee, Kouh, & Reynolds, 2013; Yamins et al., 2014), most knowledge about the tuning of V4 has derived from parameterized stimuli sets (reviewed in (Roe et al., 2012)). The increased complexity of the V4 response makes a regression approach more difficult, yet it also increases the chances that tuning curves may misrepresent the role of neurons in processing.

Our goal in this work is to develop a methodology to compare tuning between artificial and natural contexts and to discuss the implications of context-dependence. We specifically focus on tuning to hue, as this has previously been shown to strongly modulate V4 neurons yet has previously been studied by displaying simple colored shapes (Bohon, Hermann, Hansen, & Conway, 2016; Conway, Moeller, & Tsao, 2007; Conway & Tsao, 2009; Li, Liu, Juusola, & Tang, 2014; Tanigawa, Lu, & Roe, 2010). Since some degree of context-dependence may be expected for hue but the exact degree is difficult to predict, this provides an important demonstration. Using a progression of encoding models, including one based on a deep artificial network pretrained to classify images (Yamins et al., 2014), we estimated the hue tuning curves of neurons from their responses to natural scenes and then again by varying the hue of simple stimuli. Overall we found that, although hue strongly modulates the V4 response, the tuning curves estimated from responses to stimuli of a single hue poorly described how hue affected responses to natural scenes. To interpret this finding we show how such mixed selectivity may be useful for information transmission.

Materials and Methods

Experimental setup: recordings

We recorded from 96-electrode Utah arrays (1.0 mm electrode length) implanted in visual area V4. At the time of the experiment, Monkey 1 (M1) was aged 5 years, 10 months and Monkey 2 (M2) was aged 9 years, 4 months. Surgical details describing the implantation method can be found in previous publications. The array was located in the left hemisphere for monkey M1 and in the right hemisphere for M2. Spikes were sorted off-line first with an automated clustering procedure (Shoham, Fellows, & Normann, 2003) and then refined by hand using MATLAB software taking into account waveform shape and interspike interval distributions (Kelly et al., 2007).

All experimental procedures were approved by the Institutional Animal Care and Use Committee of the University of Pittsburgh.

Artificial stimuli

Both monkeys viewed uniform images of a single hue on a computer screen at 36 cm distance, with a resolution of 1024x768 pixels and a refresh rate of 100 Hz on a 21” cathode ray tube display. We found that full-field hues elicited strong and selective responses from a majority of neurons (see Results). The hues were sampled from the hue wheel in CIELUV color space (calculated with a D65 standard illuminant and standard observer) at increments of 1 degree and at a chromaticity ensured to lie in the RGB gamut, and were presented in random sequence. Monkey M1 freely viewed the stimuli, and was rewarded periodically for maintaining eye position on the screen for 4 seconds, after which time the static image was refreshed. The trial was ended if the monkey looked beyond the screen during this duration. Monkey M2 was trained to fixate a small dot at the center of the screen for 0.3 seconds, during which three images were flashed for 100ms each. A 0.5 second blank period interspersed each fixation. Monkey 1 viewed 7,173 samples of the uniform hue stimuli over 10 sessions, while Monkey 2 viewed 1,119 samples during a single session. The full monitor subtended 55.5 degrees of visual angle horizontally and 43.1 degrees vertically. The monitor was calibrated to linearize the relationship between input luminance and output voltage using a lookup table. This calibration was performed for grayscale images, and the color profile of the monitor was not separately calibrated.

Natural images

Both monkeys viewed samples from a dataset of 551 natural images, obtained from a custom-made Google Images web crawler that searched and downloaded images based on keywords such as cities, animals, birds, buildings, sports, etc. Monkey M1 viewed images over 15 separate sessions, for a total of 77961 fixations. Monkey M2 viewed images over two sessions on a single day, for a total of 6713 fixations. We then extracted the features from the image patch centered around each fixation that would serve as model inputs. The image patch around fixation corresponded to the 400 x 400 pixel block surrounding the center of gaze. This corresponds to a region 23.5 visual degrees on a side.

Gaze tracking and fixation segmentation

We employed a free-viewing paradigm for one monkey (M1) and a fixed-gaze paradigm for the other (M2). The location of each monkey's gaze on the screen was tracked with an Eyelink 1000 infrared tracker (SR Research, Ottawa, Ontario, Canada). Visual stimuli were presented and the experimental trials were controlled by custom MATLAB software in conjunction with the Psychophysics Toolbox (Brainard & Vision, 1997). For monkey M1, we segmented each fixation as a separate event based on thresholding the position and velocity of the gaze coordinates. We did not analyze activity occurring during eye movements. Once each fixation was separated, the average location of the fixation was recorded and matched to image coordinates. Monkey M2 was trained to fixate on a dot positioned at the center of each image. The gaze was tracked as for M1, but this time only to enforce fixation and terminate the trial if the gaze shifted away from center.

Session concatenation

Although all recordings in M1 were performed with the same implanted Utah array, they were recorded over several sessions. The recordings for M2 were made in a single session. In M1, this introduced the possibility that the array might have drifted, and that a single channel might have recorded separate neurons in different sessions. To address this possibility, we noted that spikes identified in a channel in one session will be less predictive of another session's activity if the neurons are not the same, as we expect tuning to be relatively static across days (Bondar, Leopold, Richmond, Victor, & Logothetis, 2009; McMahon, Jones, Bondar, & Leopold, 2014). We thus filtered out neurons whose uniform hue tuning changed across sessions. We trained a gradient boosting regression model with Poisson targets to predict spike counts in response to the hue of the stimuli. Nuisance parameters, such as duration of stimulus, gaze position, inter-trial interval, etc., were also included as model covariates to increase the predictive power even for neurons that were not hue-tuned. We then labeled a neuron as having static tuning as follows. First, we trained the model on each single session in a 8-fold cross-validation procedure and recorded the mean pseudo-R^2 score. This score reflected how well the model could predict held-out trials on the same session. Then, we re-trained the model on each session and predicted on a different session, for all pairs of sessions. This resulted in a cross-prediction matrix with diagonal terms representing same session predictability (the 8-fold CV score), and off-diagonal terms representing generalization between sessions. We did not concatenate sessions if hue tuning estimated in one session could not predict hue responses in another session (i.e. the CV pseudo-R^2 score was less than 0).

The natural image sessions were interspersed with the artificial sessions. If a natural image session occurred between two artificial sessions, and a neuron showed static tuning both artificial sessions as identified in the above manner, then that natural image session was included for the hue tuning comparison and model fitting. The recordings of units from other natural image sessions were not used. This procedure improved our confidence that the neurons recorded in different sessions were the same.

Uniform hue tuning curve estimation

Hue tuning curves were built for each neuron by plotting its spike rate on each fixation against the observed hue. Spike rates were calculated from activity 50ms after fixation onset until 300ms or fixation offset, whichever came first. For the visualizations in the figures, we performed LOWESS smoothing, in which each point of the curve is given by a locally-weighted linear regression model of a fraction of the data. The error envelope of the curve represents the 95% confidence interval given by bootstrapping over individual fixations. To calculate the correlation between tuning curves, we did not correlate the LOWESS-smoothed curves but rather the simple binned averages. We created 16 bins of hues and calculated the average spike rate for all stimulus presentations of those hues, then correlated the 16-dimensional tuning curve vector with the natural image tuning curves.

To see how well simple tuning could explain the V4 response, we interpreted these tuning curves as models of the natural image (presented in Results in Figure 2B). This was a linear model whose coefficients are set from the uniform field tuning curve. Prediction was performed such that an image patch that was all a single color would result in a prediction that was the firing rate observed in the uniform field condition, and mixtures of colors would predict linear combinations of the corresponding observed firing rates. More precisely, the predicted firing rate was a dot product of the tuning curve with the (normalized) hue histogram. To extract these hue histograms from image patches, we calculated the hue angle of each pixel in the receptive field during a given stimulus presentation (see Receptive Field estimation below) in CIELUV space. We then binned these hues into histograms with 16 bins of hues, and these histograms served as the representation of hue input to the model. The final predicted response was then added to a constant term to account for the difference in mean firing rate across contexts.

Natural scene models

We fit several models of the V4 response to natural scenes. Each of the models described below differs in the form of the encoding and the manner by which hue tuning curves are reconstructed.

Hue models

Our first model describes neural activity as a function of the hues present in the receptive field on each fixation. We used the same extraction of hues as above: we calculated the hue of pixels in the receptive field in CIELUV and binned these hues into histograms with 16 bins of hues. Since the hue histograms have 16 bins, the base regression problem to describe neural activity from hue is 16-dimensional.

As additional controls we included as covariates a small number of features unrelated to the images. To account for possible stimulus adaption, we included the trial number in the session and also the number of times the monkey previously fixated on that image. While all models predict the spike rate, which is already normalized by the fixation duration, we included the fixation duration as an input to control for possible nonlinearities of rate with fixation duration. We also included the duration of the saccade previous to the current fixation, the duration of the saccade after fixation, the location of the fixation, the maximum displacement of the gaze position during the entire duration of the fixation, and whether the pupil tracking was lost (often due to a blink) in the saccade before or after fixation. Including these inputs allowed the nonlinear methods to control for factors which also may affect spike rate.

For our nonlinear model, we selected the machine learning method of gradient boosted decision trees as implemented by XGBoost, an open-source Python package (Chen & Guestrin, 2016b). This method allows a Poisson loss function and has previously been shown to be effective in describing neural responses (Benjamin et al., 2018). Briefly, XGBoost trains multiple decision trees in sequence, with each trained on the errors of the previous trees. We chose several regularization parameters using Bayesian optimization for a single neuron. These parameters included the number of trees to train (200), the maximum depth of each decision tree (3), the data subsampling ratio (0.5), the minimum gain (0.3), and the learning rate (0.08). The generalized linear model (GLM) presented in Extended Data Fig. 3-1 was a linear-nonlinear model with an exponential link function and a Poisson loss. We included elastic net regularization, and selected the regularization coefficient for each neuron using cross-validation with k=8 folds in an inner loop in the outer cross-validation for model scoring (see Model Scoring and Cross-Validation). We implemented this with the R package r-glmnet (J. Friedman, Hastie, & Tibshirani, 2010).

To build tuning curves from hue model we predicted the response to a vector indicating which color was present (that is, a "one-hot" vector with one entry per bin of hues that is all zeros except for the hue that is present). Then, to estimate the measurement error of the tuning curves, we refit the models to the original neural responses resampled with replacement (see Calculation of Error Bound). This resulted in tuning curves from hundreds of bootstrapped model fits. In figures in which we display the tuning curves, the lower and upper error bounds represent the 5th and 95th percentiles of the tuning curves observed when refitting the models to the resampled data.

CNN model

Our convolutional neural network (CNN) encoding model was based on findings that the intermediate layers of pretrained networks are highly predictive of V4 responses (Yamins et al., 2014). Ours was built from the VGG16 network, which is a large convolutional network trained to classify the images from the ImageNet dataset (Simonyan & Zisserman, 2014). It contains 13 convolutional layers and 3 fully connected layers. We built an encoding model for each neuron from the activations of layer 14 (the first fully-connected layer), which we found to have the highest predictive power in conjunction with a nonlinear readout. We did not modify or refit this CNN to predict neural responses. Instead, we ran nonlinear Poisson regression (XGBoost) to predict each neuron's response to an image from the values of layer 14 when the VGG network was given the same image. The final model thus takes a fixation image as input, runs the image through 14 layers of the VGG16 CNN, and then through a trained instance of XGBoost to predict the spike rate of a neuron. We call the combination of the CNN model and the trained XGBoost for each neuron the “CNN model”.

The CNN model could then be used to build tuning curves. We conceptualized this as extracting the average first-order effect of hue upon the responses of this model to natural images. We perform the following cross-validated procedure for each of 8 bins of hues. First, we train the CNN model (i.e. train the XGBoost regressor) on the training set of the natural image dataset. We then modify the test set images by slightly desaturating all pixels whose hue lies within the current hue bin. The bins were chosen to be large (8 in instead of 16) to so as to be less affected by pixel noise and to speed computation. We desaturated by multiplicatively reducing the chroma of colors in LUV color space, the same color space in which we define hue, by a certain factor. For robustness, we modified images at each of many desaturation levels, ranging from 5% to 100% of their original chroma. We then obtained the predictions of the CNN model to the original test set and also for each modified, desaturated test set, and take the average difference of these two predictions across all images. This process is repeated in an 8-fold cross-validation procedure, so that each image serves as the test set once. The resulting series of average differences can be plotted against the desaturation. The slope of this line represents the average first-order contribution of that bin of hues to the images in the dataset. Note that the value of slope reflects the scale the x-axis, which represents the parameterization of the desaturation percentage. It is best to think of the units of slope as arbitrary; the important result is the relative value of the slope between hues. Finally, the process was repeated for each bin of hues, resulting in the tuning curve to hue.

We sought to validate this procedure on simulated data (Extended Data Fig. 4-1). One important aspect is that predictions are made on images that are as close to the distribution of images in the training set as possible. Since images in which a single bin of hues are desaturated by 5% are visually indistinguishable from the originals, this is not likely to be a concern. Nevertheless, we observed whether this method would be able to reconstruct the hue tuning of simulated neurons. We constructed 20 simulated neurons that responded linearly to the hues present in a receptive field. Each neuron was cosine tuned with a randomly selected hue angle. Linear regression could perfectly reconstruct the hue tuning of these simulated neurons, as expected. The CNN method could also reconstruct the tuning curves, though less well than linear regression. If linear tuning curves do exist, then, the CNN method would be able to reconstruct them.

Receptive field estimation

To estimate hue tuning on natural scenes with the hue models, we needed to know which hues were present within the RF on each fixation. We mapped the RFs by presenting sinusoidal gratings at four orientations, which were flashed sequentially at the vertices of a lattice covering a portion of the visual field suggested by anatomical location of the implant. For monkey 1, this procedure identified an average RF over neurons in the implant of 5.87° in diameter (full-width at half-maximum) centered 8.94° below and 4.99° to the right of fixation, whereas for M2 we found an average RF 7.02° wide centered 7.02° below and 7.02° to the left of fixation. The location of the RFs were confirmed in the natural scene presentations as the pixel block that allowed the best predictions on held-out trials. For the hue model analyses, on each fixation we obtained the model inputs by extracting the hues present in the 50x50 pixel block (2.93° of visual angle on a side) surrounding the centroid of the RFs of each monkey.

We did not use this RF information in the CNN model, which took as input the entire image region around the fixation. Since information about spatial location preserved in the lower and intermediate layers of the CNN, the RF for any neuron can be learned. This addressed any worry that our conclusions are dependent upon the RF specification in the two hue models, and as well that the RF specification might systematically change for natural images.

Model scoring and cross validation:

We quantified how well the regression methods described neural responses by calculating the pseudo-R^2 score. This scoring function is applicable to Poisson processes, unlike a standard R^2 score (Cameron & Windmeijer, 1997). The pseudo-R^2 was calculated in terms of the log likelihood of the true neural activity $L(y)$, the log likelihood of the predicted output $L(\hat{y})$, and the log likelihood of the data under the mean firing rate $L(\bar{y})$.

$$R^2 = 1 - \frac{\log L(y) - \log L(\hat{y})}{\log L(y) - \log L(\bar{y})} = \frac{\log L(\hat{y}) - \log L(\bar{y})}{\log L(y) - \log L(\bar{y})}$$

The pseudo-R^2 is, at left, one minus the ratio of the deviance of the tested model to the deviance of the null model. It can be also be seen, at right, as the fraction of the maximum potential log-likelihood. It takes a value of 0 when the data is as likely under the tested model as the mean rate, and a value of 1 when the tested model perfectly describes the data.

We used 8-fold cross-validation (CV) when assigning a final score to the models. The input and spike data were segmented randomly by fixation into eight equal partitions. The methods were trained on seven partitions and tested on the eighth, and this was repeated until all segments served as the test partition once. We report the mean of the eight scores. If the monkey fixated on a single image more than once, all fixations were placed into the same partition. This ensures that the test set contains only images that were not used to train the model.

Calculation of error bounds

Each estimate of a tuning curve represents, in essence, a summary statistic of noisy data. To estimate error bounds on tuning curves, we relied on the nonparametric method of bootstrapping across trials, or for summary statistics of the entire neural population, additionally bootstrapping across neurons. Since the uniform field hue tuning curves used for correlations were simple averages of spike rates, binned over hue, we bootstrapped across trials to compute the confidence intervals. The natural scene tuning curves for the nonlinear hue model represented the predicted response to single hues. For these methods, we computed uncertainty bounds on their predictions to single hues by retraining the methods on resampled datasets (with replacement) and selecting the 5th and 95th percentiles of the predicted output for each bin. For the CNN method, the tuning curves were calculated from linear fits of the difference in test set predictions as a function of hue bin desaturation. The difference in predictions was noisy across images, with large changes predicted for some images but small changes predicted for other images. This noise presented as uncertainty in the linear fit to the data. The error on the CNN tuning curve, then, represented the uncertainty in the linear fit to the test set predictions.

The uncertainty on each of the tuning curves was then propagated into the correlation between the natural scene and uniform field tuning curves. This was again done through bootstrapping. For a given natural scene/uniform field correlation, we correlated the natural scene and uniform field tuning curves from hundreds of model fits upon resampled data, yielding a large distribution of correlations. We then reported the mean, 5th, and 95th percentiles of this distribution. The uncertainty of the mean across neurons included a bootstrap across the trials used to build the tuning curves for each neuron, followed by a bootstrap across neurons.

Normative analysis of mixed selectivity

When neurons are nonlinearly selective for mixtures of a feature with others (a situation leading to tuning changing with context) they are said to have nonlinear mixed selectivity. Nonlinear mixed selectivity has previously been argued to be advantageous because it increases the dimensionality of the space of possible neural responses, which allows a greater diversity of linear readouts for downstream tasks (Rigotti et al., 2013). Here we look to the optimal coding literature to find an alternative, more general justification. Our findings are summarized in Results.

A well-studied notion of optimality is that of Fisher efficient coding. In this framework the neural code is optimized to increase the Fisher Information it contains about the features of stimuli that are important for behavior. Denoting these features as $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_M\}$, and the population activity of N neurons $\boldsymbol{x} = \{x_1, x_2, \ldots, x_N\}$, the Fisher Information is a matrix defined elementwise as:

$$F(\theta)_{i,j} = \left\langle \left(\frac{\partial}{\partial \theta_i} \ln p(\boldsymbol{x}|\theta_i) \right) \cdot \left(\frac{\partial}{\partial \theta_j} \ln p\big(\boldsymbol{x}|\theta_j\big) \right) \right\rangle_{\boldsymbol{x}} .$$

Here $\langle \cdot \rangle_x$ denotes the expectation over $\boldsymbol{x}$ given sources of noise. The Fisher Information is intuitively similar to the sensitivity of a representation across all neurons to a given dimension and value of $\boldsymbol{\theta}$. We will ask what representations maximize the Fisher Information about all M encoded features.

Maximizing Fisher Information is a good measure of optimality because it describes how well any optimized decoder can read out the features $\boldsymbol{\theta}$ from the response $\boldsymbol{x}$. Note that this normative reason is the *potential quality* of the readout, rather than the *overall number* of potential linear readouts as in the work of (Rigotti et al., 2013). Following the literature on optimal coding of neural populations (N. Brunel & J. P. Nadal, 1998; Seung & Sompolinsky, 1993; Z. Wang, Stocker, & Lee, 2012), the decoding error can be bounded with the Cramer-Rao inequality (Seung & Sompolinsky, 1993):

$$\left\langle \left(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}\right)^2 \right\rangle \geq tr(\boldsymbol{F}(\boldsymbol{\theta})^{-1}).$$

This states that the reconstruction error is lower-bounded by the trace of the inverse of the Fisher Information of the neural population with respect to $\boldsymbol{\theta}$. By this metric, larger Fisher Information matrices allow lower error. The key question is then whether mixed selectivity increases the Fisher Information of the response.

Our analysis focuses on the case of neurons that fire with a mean rate $f(\boldsymbol{\theta})$ with additional independent Poisson noise. In this case the Fisher Information is a sum over neurons (Zhuo Wang, Stocker, & Lee, 2013):

$$F(\boldsymbol{\theta}) = \sum_{i}^{N} f_i(\boldsymbol{\theta})^{-1} \, \nabla_\theta f_i(\boldsymbol{\theta}) \, \nabla_\theta f_i(\boldsymbol{\theta})^T.$$

To show that mixed selectivity is advantageous in this setting, we will show that Fisher Information is larger (as measured by the trace) when neurons respond to mixtures of features rather than code for only one feature. Our approach rests on the fact that neurons with Poisson noise have lower variance at lower firing rates. We find that mixed selectivity is better in this setting because many neurons can participate in coding, rather than waiting in silence for their single feature, and because distributed coding allow lower firing rates for the same sensitivity.

Let the spike rates for the aligned response (one feature per neuron) be denoted as $f(\boldsymbol{\theta})$. A random rotation of this response is $Rf(\boldsymbol{\theta})$, where R is a random rotation matrix. We can additionally restrict R to those rotations that preserve positivity of the resultant spike rates. Since we have assumed that neurons fire with independent Poisson noise, the Fisher Information of this mixture is

$$F_M(\boldsymbol{\theta}) = \sum_i^N \frac{1}{r_i f(\boldsymbol{\theta})} \nabla_\theta r_i f(\boldsymbol{\theta})\ (\nabla_\theta r_i f(\boldsymbol{\theta}))^T.$$

Here r_i is the corresponding row of the mixing matrix. In this case the trace of the Fisher Information (our measure of coding quality) is:

$$\mathrm{tr}F_M(\boldsymbol{\theta}) = \sum_i^N \frac{1}{r_i f(\boldsymbol{\theta})} \mathrm{tr}\ (\nabla_\theta r_i f(\boldsymbol{\theta})\ (\nabla_\theta r_i f(\boldsymbol{\theta}))^T).$$

To ask if the rotation improves coding, we can ask if either of the two terms in the summand increase or decrease, on average. This is an approximation but useful for intuition. First, we find that the average of the right term does not change. This can be seen via the linearity of the trace and derivative and the cyclic property of the trace, $\sum_i \mathrm{tr}(\nabla_\theta r_i f(\boldsymbol{\theta})\ (\nabla_\theta r_i f(\boldsymbol{\theta}))^T) = \sum_i \mathrm{tr}\big(r_i \nabla_\theta f(\boldsymbol{\theta})\ (\nabla_\theta f(\boldsymbol{\theta}))^T r_i^T\big) =$
$\mathrm{tr}\big(\nabla_\theta f(\boldsymbol{\theta})\ (\nabla_\theta f(\boldsymbol{\theta}))^T \sum_i (r_i^T r_i)\big)$. Since R is a rotation, $RR^T = I$, and the average reduces to its value absent a rotation, $\mathrm{tr}(\nabla_\theta f(\boldsymbol{\theta})\ (\nabla_\theta f(\boldsymbol{\theta}))^T)$. Thus, only the average of the scalar term due to Poisson noise $\frac{1}{r_i f(\boldsymbol{\theta})}$ is affected by the rotation.

The average value of $\frac{1}{r_i f(\boldsymbol{\theta})}$ depends on the sparsity of the response. The Fisher Information is affected only by active neurons, of which there are some number L. We can approximate the average in the unrotated population with $\frac{1}{N}\sum_i \frac{1}{f_i(\boldsymbol{\theta})} \approx \frac{L}{N f_{typ}}$ where f_{typ} is the typical firing rate of the L active neurons. This will increase for any random rotation with a probability that increases with the number of neurons because the typical firing rate is almost certain to decrease with a rotation. In fact, the dot product of a random vector such as r_i with a fixed vector $f(\boldsymbol{\theta})$ concentrates around 0 in high dimensions. One way to see this is to imagine that the average firing rate over all neurons does not change after a rotation. This means the average rate of *active* neurons falls from f_{typ} to $\frac{L\, f_{typ}}{N}$ in the rotated population. Because the average firing rate of active neurons decreases, the variance of Poisson noise is smaller and the resulting population Fisher Information is higher.

Thus, the Fisher Information is larger when the response utilizes all Poisson neurons at small firing rates, as in the rotated response, than when the response is concentrated on a sparse subset, as when each neuron codes for one feature. This makes coding for single visual features a disadvantageous strategy.

Results

We recorded the spike rates of neurons in area V4 of two macaques as they viewed images on a monitor. One monkey (M1) freely viewed images as we tracked its gaze, while the gaze of the second monkey (M2) was fixed at image center during image presentation. We analyzed the responses of 90 neurons in M1 over several viewing sessions, taking care that the identity of cells on each electrode did not drift across sessions (see Methods: Session Concatenation), and in M2 recorded from 80 neurons in a single session. We then estimated tuning curves from responses to both artificial and naturalistic stimuli in order to ask if and how hue tuning generalizes.

Tuning to hue on uniform screens

We first measured hue tuning by varying the hue of a uniform flat screen (Fig. 1A). We found that most of our neurons were well-tuned to specific hues, consistent with the previous literature on hue tuning in V4 (Conway et al., 2007; Li et al., 2014; Tanigawa et al., 2010). Neurons' strong selectivity for hues evenly tiled the hue circle (Fig. 1C). We characterized the degree of modulation with hue with the Modulation Index, calculated as the peak-to-peak range of the mean-normalized tuning curve (Fig. 1D). We also characterized our ability to estimate tuning by correlating the two tuning curves estimated on each half of the trials, selected randomly and bootstrapped for confidence bounds. The confidence interval of this correlation was usually high and excluded zero for 79/90 of neurons in M1 (Fig. 1C), but only for 17/80 neurons in M2 (Fig. 5A). A general trend across analyses was that neurons in M2 were more poorly described by hue than the neurons in M1. This difference in monkeys was possibly due to the spatial heterogeneity of color responses in V4 (Conway et al., 2007; Tanigawa et al., 2010). In later analyses, we compared the hue tuning of neurons only when we could reliably estimate tuning. Thus, the placement of our electrodes in M1 and M2 identified neurons that, as characterized by stimuli of a single hue, appeared to robustly encode the hue of stimuli.

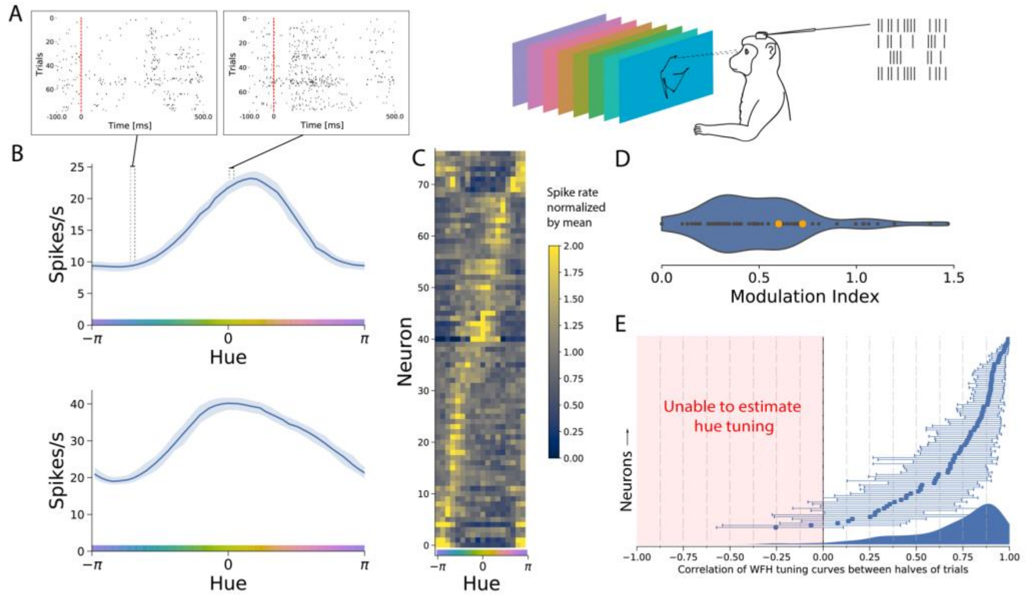


Figure 2-1:
Tuning curves estimated from responses to artificial stimuli. Data from M1; see Fig. 5 for M2. A) We recorded from neurons in area V4 as a monkey viewed fields of a uniform hue and examined the average evoked spike rate 50-300ms after presentation. B) The uniform hue tuning curves for two example neurons, showing strong hue modulation, here displayed with LOWESS smoothing of trial responses. C) Most neurons modulated their activity strongly with hue. Here the unsmoothed tuning curves (mean rate in each bin of hues) are displayed normalized by per-neuron mean firing rate for comparison. D) The degree of hue tuning can be characterized with a Modulation Index, which is the difference in min and max of the tuning curve after normalization. The two example neurons of panel B are marked in orange. E) Our ability to reliably estimate hue tuning was captured by correlating the tuning curve estimated on one half of the trials with the tuning curve estimated on the other half. This correlation would be 1 in the no-noise or infinite-data condition, and if the 95% confidence bounds from bootstrapping include zero we cannot reliably estimate tuning.

If these tuning curves capture how these neurons encode hue, these tuning curves should predict responses to other types of stimuli. For example, we might expect that if a neuron preferred uniform fields of orange hue, then that neuron would on average have higher firing rates for scenes containing predominantly orange hues. To test this, we displayed natural images in alternating sessions to the same monkeys (Fig. 2A). We found that the tuning curves were not at all informative of the natural image response. Specifically, we asked how well uniform hue tuning curves could predict natural scene responses by interpreting the curves as the coefficients of a linear response to hue, and then scoring this model (see Methods). Our scoring method is a pseudo-R^2 score which behaves roughly like an R^2 but is valid on data with Poisson noise. Scores of 1 indicate perfect prediction, scores of 0 indicate the mean firing rate is an equally good predictor, and negative scores indicate the mean rate is a better predictor of firing than the model. Of all but one neuron, the uniform field tuning curves predict natural responses with a negative pseudo-R^2 (Fig. 2B). Thus, neurons tuned for a certain color presented in isolation did not on average fire more when that color was present in natural scenes.

These observations can be explained if the hue tuning curves themselves are different between two contexts. Alternatively these neurons could respond more strongly to non-hue features that co-vary within natural images, like the visual texture of typically green plants. To distinguish these two possibilities, we next estimated tuning to hue directly from the responses to natural images and compared it with uniform hue tuning.

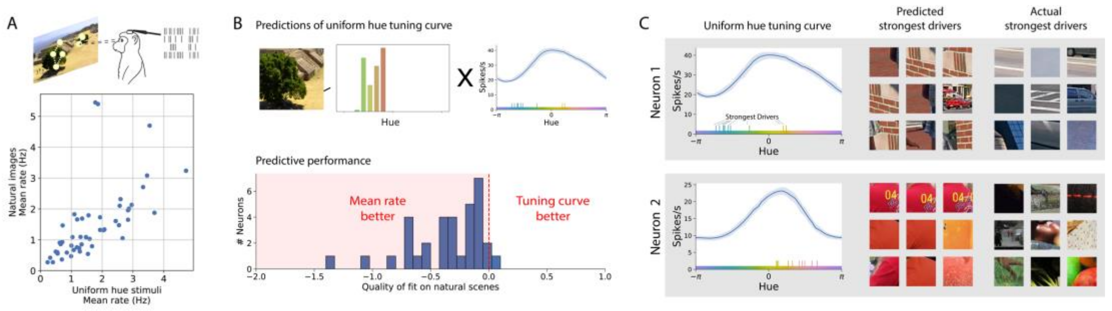


Figure 2: A) We displayed a large set of natural images to the same monkeys in interspersed sessions. Neurons fired at similar rates in these sessions as during presentations of a single hue. B) The hue-tuning on uniform hues (Fig. 1) can be treated as the coefficients of a linear model to predict neural responses to natural scenes, and scored. hue tuning on the artificial hue stimuli

could not predict any variance in the natural scene responses. This is Displayed here is the histogram of the Poisson pseudo-R^2 goodness-of-fit scores of the tuning curves' predictions, which is below zero when the predictions underperform the mean firing rate. C) The tuning curves' low predictive utility is exemplified in the disparity between the stimuli that they predicted strong responses for (e.g. the most red), and the stimuli that actually elicited the strongest responses, which were consistently of different hues.

Tuning to hue estimated from natural scenes

To investigate if hue tuning changes in the context of natural images, we directly regressed the contribution of hue to the neural response using two separate models. These models are of varying complexity and nonlinearity. Collectively they control for other visual features that drive V4 neurons, including interactions between hues, and rule out the possibility that visual confounds could explain the discrepancy between uniform field hue tuning and natural scene hue tuning.

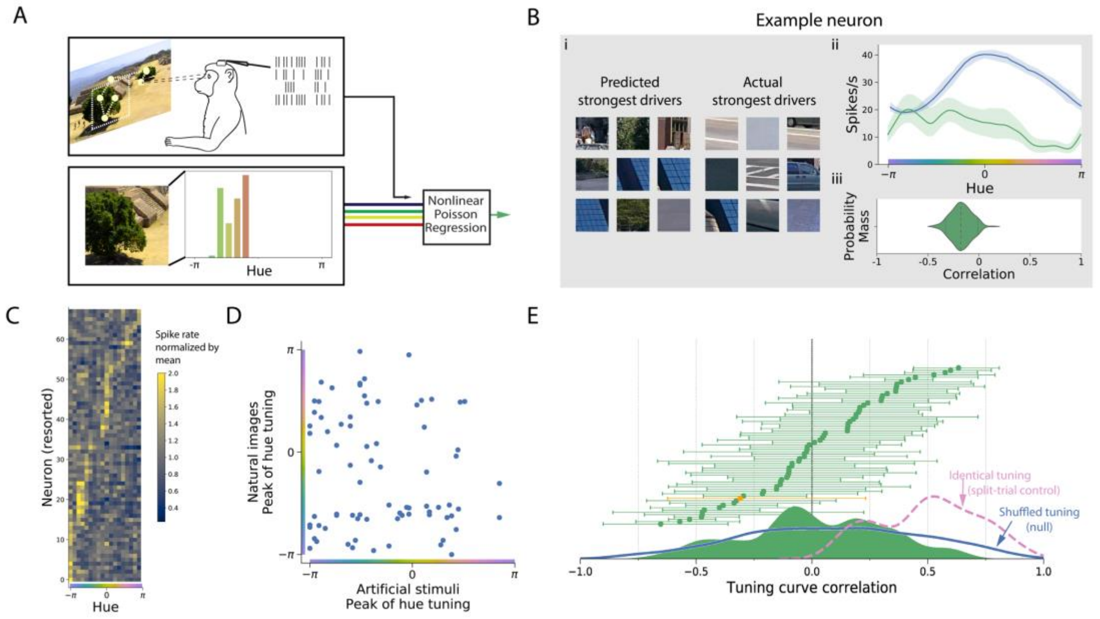


Figure 3. Tuning curves for hue estimated from the responses to natural images. Data from M1; see Fig. 5 for M2. A) We trained a nonlinear model with Poisson output to predict each neuron's response from the hues present in its receptive field during the natural scene sessions. B) (i) The 9 trials that each model predicted to have highest firing rate looked similar to trials with the actual strongest response, unlike the uniform hue model. (ii) We built tuning curves from the model by observing its response to a single hue. The uncertainty of each curve is given by the 5th and 95th percentiles of hundreds of model fits to the trials resampled with

replacement. (iii) This uncertainty is then propagated into the correlation between the uniform hue tuning curves and natural scene tuning curves. C) Tuning curves across all neurons (resorted by hue tuning in this condition). D) The peaks of these tuning curves plotted against the peaks of the tuning curves in the uniform hue condition show no circular correlation across neurons, p=0.82. E) The correlations of the natural scene and the uniform hue tuning curves on each neuron (as shown in B(ii) and (iii)) show this is not an artifact due to multiple peaks in tuning curves. The neurons are sorted by their correlation to show a cumulative distribution. The two example neurons are highlighted in orange. Below: The smoothed density of all neurons' natural scene/uniform hue correlations is similar to what would be expected if neurons randomly shuffled hue tuning between conditions (overlaid, blue). Also overlaid (in pink) is the control distribution of how the correlations might appear if tuning were the same across stimuli, which is limited by neural noise and finite trials. This is estimated conservatively by correlating tuning estimated from one half of the natural scene trials with the tuning estimated on the other half.

Nonlinear model with hue as an input feature

We first modeled natural image responses as a nonlinear function of the hues present in the receptive field of neurons during each fixation (Fig. 3A) plus control features to account for effects such as adaptation. This model, which we refer to as the 'nonlinear hue model', predicted neural activity during natural scenes quite accurately for neurons in both monkeys (Extended Data Fig. 3-1). We fit a nonlinear model because nonlinear hue interactions have been previously observed in V4 (Kusunoki, Moutoussis, & Zeki, 2006), which would lead to a bias in a generalized linear model (GLM) because hues are correlated in natural scenes (Fig. 3-1 A). Indeed, the nonlinear model was much more accurate than a GLM fit to predict spike rates using hues of natural images (Fig 3-1 B,C). This approach to estimating hue tuning directly regresses the response to bins of hues in natural images.

We estimated hue tuning curves for the nonlinear hue by measuring its responses to single hues, in essence reproducing the uniform hue experiment but on the natural scene model. These tuning curves showed clear preference for small ranges of hues (Fig. 3C). We quantified our ability to estimate hue tuning in two ways. First, we repeatedly refit the model on the natural scene trials resampled with replacement, and observed the distribution of coefficients (Fig. 3Bii). This distribution was propagated through to later analyses such as the correlation between a neuron's hue tuning estimated in either stimuli set. Secondly, we visualized how high the correlation of hue tuning across conditions would have appeared if tuning were the same in both contexts, given all sources of noise and measurement error. This we estimated by comparing the hue tuning curves from two non-overlapping halves of natural scene trials (Extended Data Fig. 2-1). Note that this split-trial control is a conservative lower bound of our quality of estimation, as the model was fit on only half the number of trials. By this measure, the nonlinear hue model was able to consistently estimate hue tuning for the most neurons in M1 (Fig 2-1 A, see also Fig. 3E) but for just two neurons in M2 (Fig. 5 D-F), which prevented a statistical analysis in M2. These estimates of uncertainty serve as a baseline limit of how well we can observe changes in hue tuning.

We compared the tuning curves across stimulus sets for neurons for which we could consistently estimate hue tuning. The peaks of the tuning curves did not show any correlation across neurons (Fig. 3D; circular correlation of -0.02, unable to reject the uncorrelated hypothesis with p=0.82). In addition, the shapes of the tuning curves did not correlate between conditions (Fig. 3E). If hue affected V4 responses in the same way in both contexts, we would have observed the correlations between tuning curves across contexts to be at least as positive as the split-trial control. This was not the case. In M1, the natural scene/uniform field tuning curve correlations were significantly lower than these split-trial correlations ($p=1.0\times10^{-14}$, Wilcoxon signed-rank test; Extended Data Fig. 2-1 D), indicating that the observed change in hue tuning across contexts was not a consequence of noise in the estimation of tuning. In fact, the spread of correlations between the two sets of tuning curves was similar to the distribution that would arise if hue tuning shifted randomly between contexts (Fig. 3E inset), which to preserve typical tuning shapes we approximated as the correlations between random neurons' tuning. Thus, the regressed contribution of hue to the neural response had little relationship to the strong tuning observed in response to stimuli of a single hue.

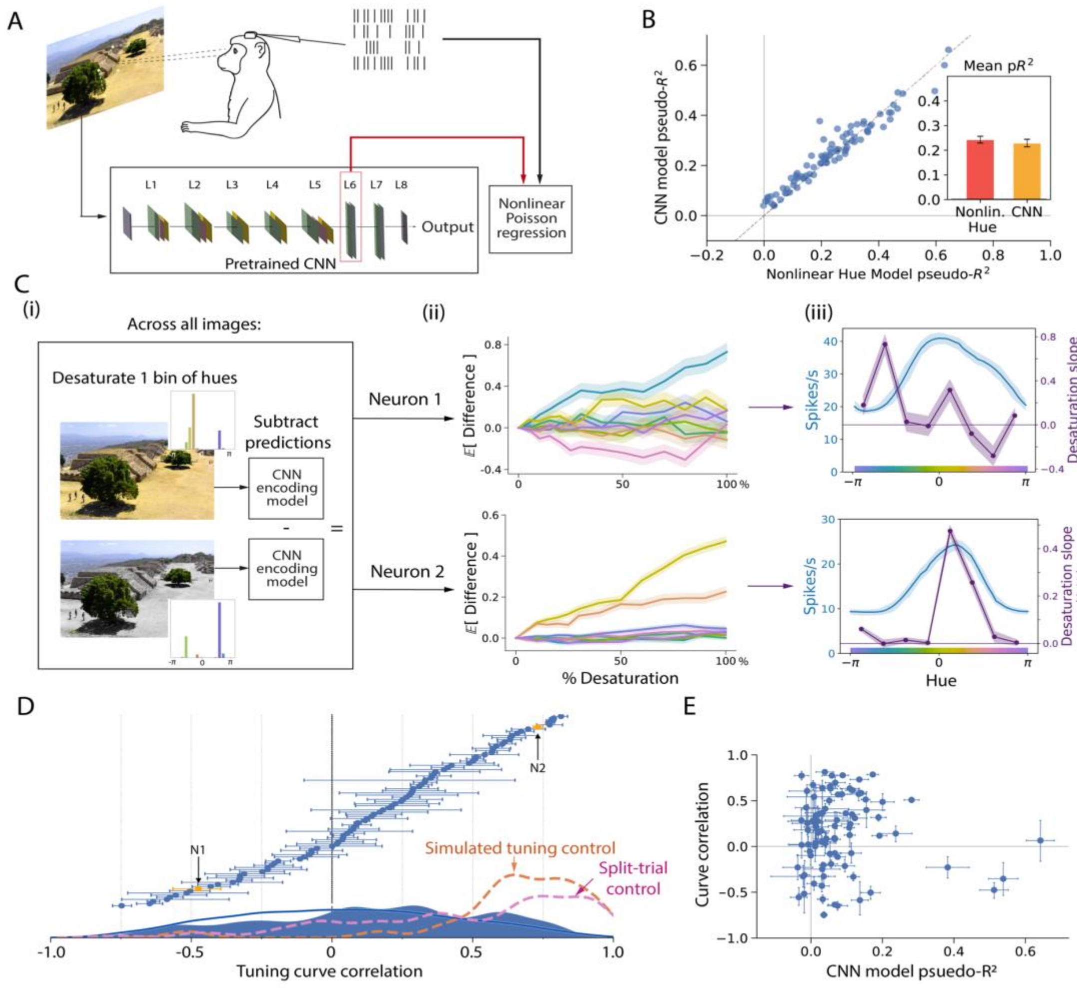


Figure 4. Tuning curves estimated for hue from a model of V4 responses built from a pretrained convolutional neural network (CNN). Data from M1; see Fig. 5 for M2. A) We trained a nonlinear Poisson regression model (gradient boosted trees) to predict the V4 response from the activations of an intermediate layer in the VGG16 network given the visual stimulus. B) The quality of the neural predictions on each neuron, measured by the cross-validated pseudo-R^2 score, were similar between the CNN model and the nonlinear hue model. C) We built hue tuning curves in the following manner: (i) For each image in a test set, we slightly desaturated all pixels in a bin of hues, and subtracted the CNN model's predictions on the perturbed image from those on the original image. (ii) For each neuron, the average change in the predicted response across all test images was plotted against the percentage by which hues were desaturated. The slope of each line is, to first order, the average effect of that hue on the model response in the test set. The top and bottom plots show the same example neurons as in earlier plots. (iii) The resulting tuning curve (purple) summarizes the average effect of each of the 8 bins of hues – i.e. the slopes of the 8 desaturation curves. It can be seen that the tuning of neuron 1 was poorly correlated with the uniform hue tuning (blue), while that of neuron 2 was well-correlated, in agreement with the hues of the strongest-driving stimuli shown in Fig. 1B. D) We calculated the correlation between

the two tuning curves for all neurons. The distribution of correlations was lower than for the reconstructed hue tuning of simulated neurons (“simulated tuning control”; see also Extended Data Fig. 4-1) as well as the distribution of correlations between tuning curves estimated from two non-overlapping halves of the natural scene trials (“split-trial control”; see also Extended Data Fig. 2-1). E) The quality of the CNN model fit for each neuron did not predict the correlation of the tuning curves.

Neural network model of V4 responses

We next repeated the estimation of hue tuning on natural scenes with a more general model of V4 neurons that does not rely on hand-specified summaries of the features present in a receptive field. This was important to ensure that our results were not sensitive to design decisions in processing the images, as well as to account for the confounds of other, non-hue features contained in the image. The two hue models would provide biased estimates of tuning if neurons also responded to other visual features, and if these features co-varied in the image dataset with hue. If most green objects are plants, for example, the observed dependence on green hues may be partially attributable to a response to the high spatial frequency of greenery. Theoretically, one could include these features as additional covariates, but the list of features that drive the V4 response in a simple manner (e.g. linearly) is not comprehensively known. Good progress has been made with shape and texture (Okazawa, Tajima, & Komatsu, 2015; Pasupathy & Connor, 2001; Portilla & Simoncelli, 2000), but arguably not enough to thoroughly control for all non-hue features in a simple model. Controlling for other drivers of V4 thus requires a model that learns relevant visual features instead of using features chosen by a researcher or parameterized by hand.

The model we selected was based on an encoding model of V4 that relates neural responses to the activations to the upper layers of a convolutional neural network (CNN) pretrained to classify images (Yamins et al., 2014). Such “transfer learning” models have also recently been used to infer optimal stimuli for neurons in V4 (Bashivan, Kar, & DiCarlo, 2019; Cowley et al., 2017). Instead of pre-specifying a receptive field estimated with sparse noise, we allowed the CNN model to learn any location sensitivity itself and thus fed the entire fixation-centered image as input. Predictions of neural activity are obtained by passing the image through the network, obtaining the intermediate network activations, and then passing these to a classifier trained for each neuron (Fig. 4A). The predictions of neural activity given by this model were comparable in accuracy to those of the nonlinear hue model (Fig. 4B) despite the model making many fewer assumptions about how raw pixels related to responses.

Our initial, unsuccessful method to estimate hue tuning from this model was to simply observe the model's response to images of a uniform hue, as before. However, this approach failed to reconstruct tuning on simulated data. This interesting parallel to our main finding is likely due to the fact that uniform field test images are far outside the domain of natural scenes on which the CNN was pretrained.

Instead, we developed a method to estimate hue tuning from the model that only uses responses to images close to the domain of natural images. By slightly perturbing the hue of input images and observing the change in the learned model's response, we could test the model's sensitivity to hues to in natural images (Fig. 4C). First, for a test set of images not used for training, we desaturated all pixels within a bin of hues by a set percentage (Fig. 4Ci). The percentage of desaturation varied from 0% (i.e. no change) to 100% (in which all pixels of one hue are taken to the isoluminant grey). We took the difference between the model's predictions on the original and perturbed images and examined how severely this difference depended on the level of desaturation (Fig. 3Cii, iii). For each neuron, we averaged over the entire image dataset to yield the average effect of perturbing each hue on natural images. This method established the effect of hue only in the tight neighborhood of each image, and is set up to estimate the average local effect of hue on the natural image response.

To ensure that this process could in principle reconstruct correct tuning curves, we built simulated responses (Extended Data Fig. 4-1). We generated random cosine tuning curves, then simulated a hue response by applying these as linear filters upon the histograms of the hues present in each image. We then attempted to predict these simulated responses from the activations of the pretrained CNN given the raw images. Using the method of progressively desaturating test images, we found we could reconstruct the original cosine tuning curves with high accuracy (Fig. 4D overlay and Fig. 4-1). As a second, more conservative test, we also performed the split-trial control for the actual V4 neurons, which involved repeating the entire analysis separately on two non-overlapping halves of natural scene trials and then correlating the two resulting tuning curves. The split-trial tuning curves showed significantly positive correlations for most neurons in M1 (Fig. 4D overlay) and M2 (Fig. 5). This method of querying the effect of hue could thus accurately estimate hue tuning curves from natural scene responses in both monkeys.

We next asked if these tuning curves would be similar to the tuning curves to uniform hues. We found that the tuning curves of one context were different from tuning in the other (Fig. 4D for M1 and Fig. 5G for M2). Among those neurons for which we could consistently estimate hue tuning, the natural scene/ uniform hue tuning curve correlations were significantly closer to 0 ($p=1.1x10^{-8}$, Wilcoxon signed-rank test, Extended Data Fig. 2-1 for M1; Fig. 2H for M2). This difference in tuning curves was not an artifact of our model fit or estimation method, as this would be measured in the split-trial control, and additionally we observed no correlation between the model's accuracy on unseen natural images and the natural scene/uniform field correlation (Fig. 4E and Fig. 5I).

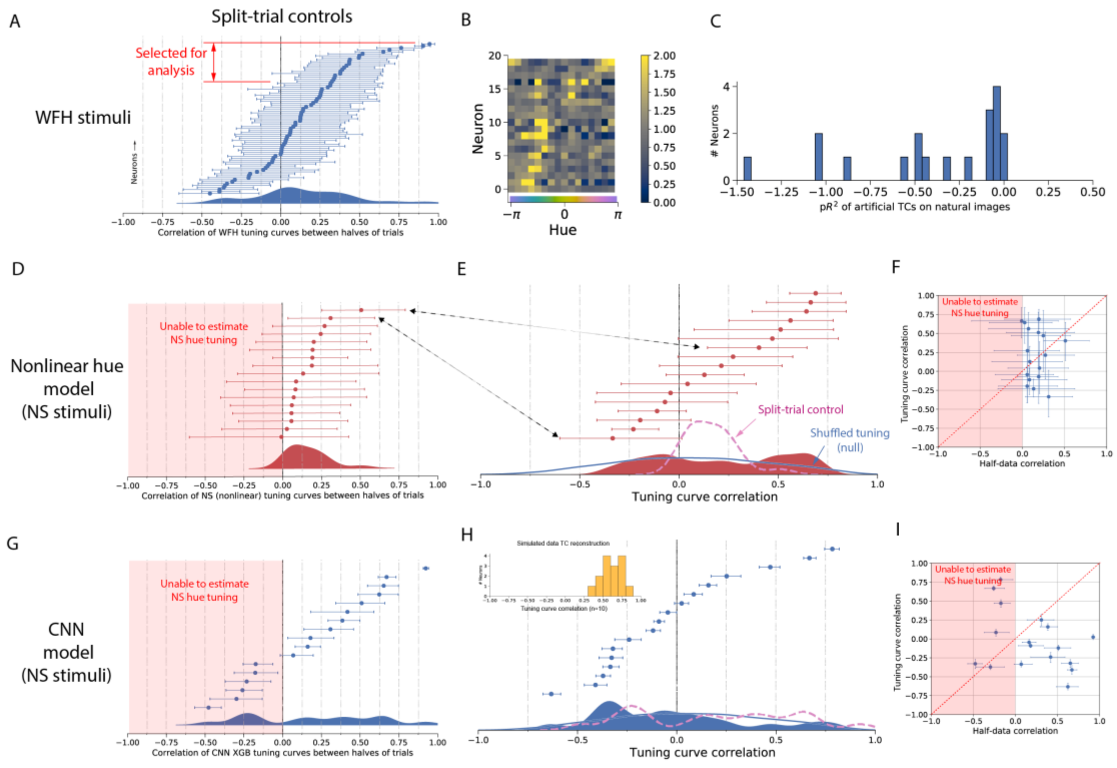


Figure 5: Collected data for M2. A) Most neurons in M2 showed poor hue tuning, and we were not able to consistently estimate uniform hue tuning nearly as well as for M1. B) Binned tuning curves for the neurons selected in A. C) As for M1, the uniform hue tuning curves were worse at predicting natural scene responses than the mean firing rate on natural scenes. D-F) Analysis of the natural scene tuning curves estimated by the nonlinear hue model was inconclusive. D) The natural scene tuning curves could not be estimated as consistently as for M1. E) The natural scene/uniform hue tuning curve correlations as estimated by the nonlinear hue model. Like for M1 (Fig. 2c) we overlay the split-trial distribution and the null distribution expected with random reshuffling of hue tuning. F) By a Wilcoxon signed rank test, we were unable to reject the null hypothesis that natural scene/uniform hue correlations are lower than the split-trial correlations (p=0.65) and thus it was not clear from the hue model on M2 neurons whether hue tuning does or does not change. G-I) Analysis of the natural scene tuning curves estimated with the CNN method. G) Distribution of correlations of hue tuning estimated of non-overlapping halves of trials. H) Natural scene/uniform hue correlations. Inserted is the distribution of natural scene/uniform hue correlations of simulated neurons with cosine hue tuning. Since M2 saw 10x fewer trials than M1, we simulated again on this smaller dataset. I) Among the neurons for which we could consistently estimate hue tuning (i.e. with a positive correlation of tuning curves estimated on split data), all neurons had a higher split-trial natural scene curve correlation than a natural scene/uniform hue correlation. This was significant under a Wilcoxon signed rank test at p=0.003.

In addition to changes in tuning curve shape as captured by correlation, we also examined if the natural scene tuning curves showed changes in the overall degree of hue modulation. We found that hue modulation – the maximum of a tuning curve minus the minimum, normalized by the mean – was related across contexts, but weakly (Extended Data Fig. 4-2). Many neurons strongly modulated by hue on uniform fields had weak responses to hue on natural scenes, and vice versa. Overall, the tuning curves estimated with this more advanced method support our previous conclusion that hue tuning on uniform fields does not agree with the effect of hue in natural scenes.

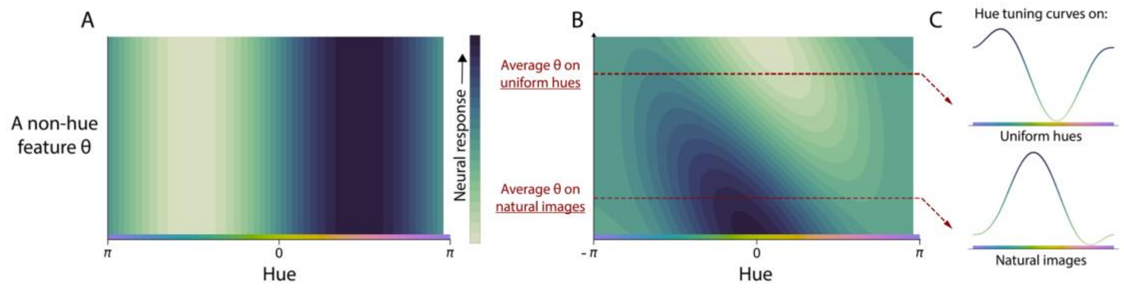


Figure 6: Interactions between features allow neurons to carry more information in their activity. A) In this two-dimensional tuning curve, a hypothetical neuron responds to only hue and carries no information about other variables. B) A hypothetical neuron that additionally responds another non-hue feature is informative about multiple dimensions of stimuli (due to its nonzero derivative). C) We can build a hue tuning curve for this neuron by varying hue with the other feature held fixed. If the average non-hue feature is different between natural images and uniform hues, the tuning curves to hue will differ between contexts.

Why features interact

A straightforward explanation of why hue tuning differs across visual contexts is that these neurons respond to nonlinear combinations between hue and non-hue features, as shown schematically in Figure 6. What computational advantage could explain this coding scheme for visual perception? It is clear that if the role of these neurons were to encode hue alone, then any nonlinear interactions would be detrimental. This is because hue can no longer be unambiguously read out without additional contextual information. Therefore these V4 neurons likely assist in a more general task, like object recognition or segmentation. Other studies have also noted that color vision may be best thought of in terms of task performance (Rosenthal et al., 2018); the absorbance spectra of the L and M photoreceptors in primates, for example, are not maximally separated as in birds but rather overlap significantly, possibly because this helps to discriminate and classify fruit and leaves (Osorio & Vorobyev, 2008). The question then arises: why would neurons being responsive to multiple features help visual processing?

One possible reason is that selectivity to multiple features increases the dimensionality of the space of possible neural responses, which allows a greater diversity of linear readouts for downstream tasks (Rigotti et al., 2013). This justification prioritizes the diversity of possible uses of an area rather than the accuracy of encoding. An additional justification can be found in the optimal coding literature (e.g. (N. Brunel & J.-P. Nadal, 1998; Seung & Sompolinsky, 1993; Zhuo Wang et al., 2013)). Starting with a certain set of behaviorally-relevant visual features, what is the optimal way of representing these M features in a single population of N neurons?

Our findings, derived in Methods, show that mixed selectivity allows for better neural codes when neurons fire as Poisson processes and visual features are sparse and often not present. This is because a mixed selectivity strategy allows many more neurons to participate in each response. When each neuron responds to k features, k times more neurons can respond on average to each scene. Crucially, a distributed response in turn enables lower firing rates for the same sensitivity, which is advantageous because Poisson noise has lower variance at low firing rates. Coding quality, which is related both to the variance of internal noise and the sensitivity of the response when measured by Fisher Information, thus improves when the response is maximally distributed across many neurons at low firing rates. Recent studies using other measures of coding besides Fisher Information support this conclusion that mixed selectivity improves neural encoding (Johnston, Palmer, & Freedman, 2020).

It should be noted that an optimal coding argument may not necessarily explain selectivity to all features. Orientation selectivity, for example, is similar across contexts (Touryan et al., 2005) and this may be related to fundamental visual cortical architecture and the importance of visual form for behavior. Where neurons do not show mixed selectivity, anatomical or behavioral constraints may override mixed selectivity's benefits of increased precision and quality of the neural encoding.

Discussion

For populations of V4 neurons in two macaques, we found that varying the hue of simple stimuli produced tuning curves that do not accurately describe hue tuning measured from natural scenes. While some discrepancy may be expected, we found that the two sets of tuning curves correlated not much better than chance. This finding was robust across multiple methods of estimating tuning, which together accounted for the confounds of both hue-hue interactions as well as of non-hue drivers of V4 activity. A hue tuning curve for V4 estimated from any one set of stimuli thus does not universally describe the average response to hue on other stimuli.

Known sources of modulation in visual responses

The V4 response is modulated by a number of factors that change with visual context. These factors are divided in the manner of their relevance to our findings. First are possible reasons why we might have observed low tuning curve correlations even if, in fact, tuning did not change between contexts. The second category of factors are known interactions between hue and other features in the V4 response that may explain why hue tuning in V4 changes with visual context. We will review both in turn.

Neurons in V4 are have been shown to preferentially respond to objects near the center of attention, even when attention falls away from fixation (Connor, Gallant, Preddie, & Van Essen, 1996; Connor, Preddie, Gallant, & Van Essen, 1997; Gallant, Connor, Rakshit, Lewis, & Van Essen, 1996). This phenomenon of receptive-field remapping is most problematic for our hue models, which required that we extract the hues lying within the receptive field. If the monkeys' attention frequently strayed away from fixation, we would have extracted hues from an irrelevant image portion. This would introduce some noise in the hue covariates, and therefore some smoothing of hue tuning curves. The CNN model learned any spatial sensitivity directly from the natural scene responses. However, the effect of attention upon receptive fields could not be modeled and it is likely that some smoothing of the hue tuning curve occurred for this technique as well. Smoothing would obscure fine-scale structure in the tuning curves. As the curves were already smooth, however, the natural scene/uniform field correlations should not be much diminished.

The smoothing effect is furthermore not consistent with our finding that many neurons have natural scene hue tuning with zero, or even negative correlation with their uniform field tuning while still showing strong hue-dependent modulation.

We now turn to potential descriptions of the interactions that might have led to a shift in hue tuning across contexts. One possibility is the behavioral phenomenon of color constancy, which would present as a neural correlate as responses to the inferred surface color of objects rather than their apparent color (which reflects the color of ambient light) (Kusunoki et al., 2006). This is a clear example of the nonseparability of the V4 response to hue, and a reason hue tuning might change between any two, single stimuli. It is less obvious, however, that color constancy correlates would cause the average effect of hue over all natural images to be different than on uniform hues. It would be expected that over tens of thousands of images with a broad range of lighting conditions, color constancy would result in some smoothing of the estimated tuning curve due to the difference between the pixels' hue and the inferred hue, and of the same characteristic scale as their typical difference. Additionally we may expect a bias that would result from the discrepancy between pure white and the average lighting condition. We expect this discrepancy to be small, and therefore that natural scene tuning curves would still be strongly (though not perfectly) correlated with the uniform field tuning curves. Though phenomena like color constancy would affect hue tuning on natural scenes, it cannot account for the entire difference we observed, and it is likely that there exists other undocumented sources of nonseparability.

Another factor is visual attention (Chelazzi, Della Libera, Sani, & Santandrea, 2011; David, Hayden, Mazer, & Gallant, 2008). A particularly relevant form of attention is feature-based attention, in which neurons tuned for a feature (say, red) increase their firing rate if that feature is attended to (as in the task, "find the red object") (Mirabella et al., 2007; Motter, 1994). While the task of M1 was free viewing and involved no instructions, it is likely that the monkey's attention shifted during the task and that it was influenced by object salience. This would affect apparent tuning if object salience were correlated with hue. Attention is less likely to have presented a confound in the task of M2, in which gaze was fixed at center and stimuli were presented for 100ms.

A subpopulation of neurons in V4, so-called equiluminance cells, respond to object boundaries defined solely by chromatic boundaries (Bushnell, Harding, Kosai, Bair, & Pasupathy, 2011). Such shapes are defined by changes in hue or saturation, and so it is worth asking whether the response function of equiluminance cells includes interactions between hue/saturation and spatial arrangement. However, it was not originally determined if the responses were actually separable in this way, as neurons' hue tuning curves were characterized with a fixed shape. It is possible that equiluminant cells had fixed hue tuning that was then modulated by shape. Thus, it is plausible but undetermined that equiluminance cells would show different hue tuning across shape and explain our results.

The apparent shift in hue tuning in natural scenes may be partially be explained by a multiplicative or generally nonlinear interaction between shape and color, as was examined in a recent paper that jointly varied the hue and shape of simple stimuli (Bushnell & Pasupathy, 2012). In a linear model with terms for hue and a multiplicative interaction between hue and shape parameters, the authors observed a significant interaction between shape and color in the majority of cells (44/60). This interaction would cause (linear) hue tuning to appear different for natural images with varying shapes, as we observe. We note that other, undescribed features may also interact with hue, and that conclusively determining which visual features interact would require presenting stimuli tiling many more dimensions of variation.

Implications for V4 and for the tuning curve approach

Color responsivity has long been a defining feature of V4 (S. Zeki, 1980; S. M. Zeki, 1973). Recent studies have shown that localized areas in V4 are strongly responsive to color (Conway et al., 2007), and furthermore that the anatomical organization of color preference on the neocortex is similar to perceptual color spaces (Bohon et al., 2016; Conway & Tsao, 2009; Li et al., 2014). These findings have been taken as evidence that areas within V4 are specialized for the perception of color. However, each of these studies characterized hue tuning by changing the color of simple shapes. Since the color tuning of V4 neurons changes with visual context, as we show here, it is possible that previous conclusions about the functional organization of V4 do not accurately describe how V4 processes more naturalistic stimuli.

It should be noted that our simple stimuli were not colored shapes chosen by hand to drive V4 neurons strongly, as in many previous studies, but rather uniform screens. We found that these still elicited strong responses and well-identified tuning curves. Nevertheless it may be objected that previous studies' stimuli may still measure hue tuning that generalizes to natural scenes. However, this would require that the factors that modulate hue tuning in V4 neurons are only present in uniform screens. It is more parsimonious that hue-tuned neurons are modulated by interactions with a range of spatial features, which collectively will cause tuning on any set of stimuli to not generalize to naturalistic stimuli.

Based on the discovery of robust tuning for the color of simple visual stimuli, some studies have concluded that the role of color-responsive areas in V4 is to represent color. Our results do not rule this out; for example these areas might represent color but be modulated by what colors are likely given the surroundings. This would complicate a read-out of color from V4, but may have other advantages like efficiency. It would be interesting to investigate this possibility in future studies. An alternative possibility is that the color-sensitive areas of V4 are not specialized to represent color, *per se*, but rather serve a more complex role within recognition and perception. This is analogous to how V2 appears tuned to orientation but can perhaps be better described as processing naturalistic texture (Ziemba, Freeman, Movshon, & Simoncelli, 2016). Furthermore, this role aligns with the suggestion that the ventral temporal cortex at large decomposes scenes into neural activity such that object categories are linearly separable (Grill-Spector & Weiner, 2014). Thus, the color-responsive areas of V4 may represent how color informs an inference of object identity (Rosenthal et al., 2018). Whether the color responses of V4 are an end to themselves (i.e. representing color) or intermediate computations in a larger assessment of object identity (DiCarlo & Cox, 2007), or both, cannot be decided from this study; both are consistent with the data.

Our study joins a longer history of literature observing that, across many brain areas, tuning curves previously characterized with simple stimuli in fact change with context. In

V1, for example, researchers found that receptive fields change with certain visual aspects that were not varied within previous stimuli sets, such as the presence of competing orientations (Fitzpatrick, 2000; Heeger, 1992; Knierim & Van Essen, 1992; Sillito & Jones, 1996). Even sound has been shown to modulate V1 receptive fields, at least in mice (McClure Jr & Polack, 2019). More recently, it was observed that receptive fields are different in the contexts of dense versus sparse noise for neurons in layer 2/3 of V1 (Yeh, Xing, Williams, & Shapley, 2009). Spatio-temporal receptive fields of V1 neurons also appear different when estimated on natural movies versus drifting gratings (David & Gallant, 2005; David et al., 2004) (though note that orientation tuning is similar for static natural scenes versus gratings (Touryan et al., 2005)). In other areas, contextual modulation has been identified by showing perturbed natural images instead of white noise (Goldin et al., 2021; Heitman et al., 2016; McIntosh, Maheswaranathan, Nayebi, Ganguli, & Baccus, 2016b) or by comparing the performance of a model that assumes separability (such as a GLM) with a nonlinear model that does not (Benjamin et al., 2018). Thus, while tuning curves generalize in some situations (e.g. (Touryan et al., 2005)), it is common that they do not, and any assumption of separability of the neural response should be verified. Furthermore, as derived in Methods, feature interactions are likely optimal for visual processing when the full visual scene is represented in neural activity and should be expected. Unless specifically investigated, it might not be correct to assume that a tuning curve accurately describes the neural response on different stimuli than used to create it.

If it cannot be assumed that neural tuning is separable, however, it becomes necessary to test prohibitively many stimuli or else make an alternative simplifying assumption. Since the number of tested stimuli must follow the number of potential feature combinations, the overall number of stimuli will grow exponentially with the number of features. When there are very many features, even very large recording datasets by today's standards may be insufficient.

One possible way forward is to make simplifying assumptions, i.e. to set strong priors of the kinds of tuning curves that could be expected. This is the approach taken, for example, when modeling neurons using the activations of deep neural networks pretrained on image classification tasks (Ponce et al., 2019; Yamins et al., 2014) or considering neural responses as implementing a sparse code (Felsen, Touryan, & Dan, 2005; Vinje & Gallant, 2000). To compare with the previous literature, single dimension experiments can then be performed on these complex encoding models, as we demonstrate here, or alternatively performed directly on artificial neural networks to gain intuition about what tuning curves say about information processing (Morcos, Barrett, Rabinowitz, & Botvinick, 2018; Pospisil, Pasupathy, & Bair, 2018). In general, finding suitable priors will require the use of strong theoretical ideas and mechanistic hypotheses. To estimate tuning without assuming separability, then, neurophysiology must embrace and develop theories of neural processing.

References

Bashivan, P., Kar, K., & DiCarlo, J. J. (2019). Neural population control via deep image synthesis. *Science, 364*(6439), eaav9436.

Benjamin, A. S., Fernandes, H. L., Tomlinson, T., Ramkumar, P., VerSteeg, C., Chowdhury, R. H., . . . Kording, K. P. (2018). Modern machine learning as a benchmark for fitting neural responses. *Frontiers in Computational Neuroscience, 12*.

Bohon, K. S., Hermann, K. L., Hansen, T., & Conway, B. R. (2016). Representation of perceptual color space in macaque posterior inferior temporal cortex (the V4 Complex). *Eneuro, 3*(4), ENEURO. 0039-0016.2016.

Bondar, I. V., Leopold, D. A., Richmond, B. J., Victor, J. D., & Logothetis, N. K. (2009). Long-term stability of visual pattern selective responses of monkey temporal lobe neurons. *PLoS ONE, 4*(12), e8222.

Brainard, D. H., & Vision, S. (1997). The psychophysics toolbox. *Spatial vision, 10*, 433-436.

Brunel, N., & Nadal, J.-P. (1998). Mutual information, Fisher information, and population coding. *Neural computation, 10*(7), 1731-1757.

Brunel, N., & Nadal, J. P. (1998). Mutual Information, Fisher Information, and Population Coding. *Neural computation, 10*(7), 1731-1757. doi:10.1162/089976698300017115

Bushnell, B. N., Harding, P. J., Kosai, Y., Bair, W., & Pasupathy, A. (2011). Equiluminance cells in visual cortical area V4. *Journal of Neuroscience, 31*(35), 12398-12412.

Bushnell, B. N., & Pasupathy, A. (2012). Shape encoding consistency across colors in primate V4. *Journal of Neurophysiology, 108*(5), 1299-1308.

Cameron, A. C., & Windmeijer, F. A. (1997). An R-squared measure of goodness of fit for some common nonlinear regression models. *Journal of Econometrics, 77*(2), 329-342.

Carandini, M., Demb, J. B., Mante, V., Tolhurst, D. J., Dan, Y., Olshausen, B. A., . . . Rust, N. C. (2005). Do we know what the early visual system does? *Journal of Neuroscience, 25*(46), 10577-10597.

Chelazzi, L., Della Libera, C., Sani, I., & Santandrea, E. (2011). Neural basis of visual selective attention. *Wiley Interdisciplinary Reviews: Cognitive Science, 2*(4), 392-407.

Chen, T., & Guestrin, C. (2016b). *Xgboost: A scalable tree boosting system.* Paper presented at the Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining.

Connor, C. E., Brincat, S. L., & Pasupathy, A. (2007). Transformation of shape information in the ventral pathway. *Current Opinion in Neurobiology, 17*(2), 140-147.

Connor, C. E., Gallant, J. L., Preddie, D. C., & Van Essen, D. C. (1996). Responses in area V4 depend on the spatial relationship between stimulus and attention. *Journal of Neurophysiology, 75*(3), 1306-1308.

Connor, C. E., Preddie, D. C., Gallant, J. L., & Van Essen, D. C. (1997). Spatial attention effects in macaque area V4. *Journal of Neuroscience, 17*(9), 3201-3214.

Conway, B. R., Moeller, S., & Tsao, D. Y. (2007). Specialized color modules in macaque extrastriate cortex. *Neuron, 56*(3), 560-573.

Conway, B. R., & Tsao, D. Y. (2009). Color-tuned neurons are spatially clustered according to color preference within alert macaque posterior inferior temporal cortex. *Proceedings of the National Academy of Sciences, 106*(42), 18034-18039.

Cowley, B., Williamson, R., Clemens, K., Smith, M., & Byron, M. Y. (2017). *Adaptive stimulus selection for optimizing neural population responses.* Paper presented at the Advances in Neural Information Processing Systems.

David, S. V., & Gallant, J. L. (2005). Predicting neuronal responses during natural vision. *Network: Computation in Neural Systems, 16*(2-3), 239-260.

David, S. V., Hayden, B. Y., Mazer, J. A., & Gallant, J. L. (2008). Attention to stimulus features shifts spectral tuning of V4 neurons during natural vision. *Neuron, 59*(3), 509-521.

David, S. V., Vinje, W. E., & Gallant, J. L. (2004). Natural stimulus statistics alter the receptive field structure of v1 neurons. *Journal of Neuroscience, 24*(31), 6991-7006.

DiCarlo, J. J., & Cox, D. D. (2007). Untangling invariant object recognition. *Trends in cognitive sciences, 11*(8), 333-341.

Felsen, G., & Dan, Y. (2005). A natural approach to studying vision. *Nature Neuroscience, 8*(12), 1643.

Felsen, G., Touryan, J., & Dan, Y. (2005). Contextual modulation of orientation tuning contributes to efficient processing of natural stimuli. *Network: Computation in Neural Systems, 16*(2-3), 139-149.

Fitzpatrick, D. (2000). Seeing beyond the receptive field in primary visual cortex. *Current Opinion in Neurobiology, 10*(4), 438-443.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software, 33*(1), 1.

Gallant, J. L., Connor, C. E., Rakshit, S., Lewis, J. W., & Van Essen, D. C. (1996). Neural responses to polar, hyperbolic, and Cartesian gratings in area V4 of the macaque monkey. *Journal of Neurophysiology, 76*(4), 2718-2739.

Gallant, J. L., Connor, C. E., & Van Essen, D. C. (1998). Neural activity in areas V1, V2 and V4 during free viewing of natural scenes compared to controlled viewing. *Neuroreport, 9*(9), 2153-2158.

Goldin, M. A., Lefebvre, B., Virgili, S., Ecker, A., Mora, T., Ferrari, U., & Marre, O. (2021). Context-dependent selectivity to natural scenes in the retina. *bioRxiv.*

Grill-Spector, K., & Weiner, K. S. (2014). The functional architecture of the ventral temporal cortex and its role in categorization. *Nature Reviews Neuroscience, 15*(8), 536.

Heeger, D. J. (1992). Normalization of cell responses in cat striate cortex. *Visual neuroscience, 9*(2), 181-197.

Hegdé, J., & Van Essen, D. C. (2003). Strategies of shape representation in macaque visual area V2. *Visual neuroscience, 20*(3), 313-328.

Heitman, A., Brackbill, N., Greschner, M., Sher, A., Litke, A. M., & Chichilnisky, E. (2016). Testing pseudo-linear models of responses to natural scenes in primate retina. *bioRxiv*, 045336.

Hung, C. P., Kreiman, G., Poggio, T., & DiCarlo, J. J. (2005). Fast readout of object identity from macaque inferior temporal cortex. *Science, 310*(5749), 863-866.

Johnston, W. J., Palmer, S. E., & Freedman, D. J. (2020). Nonlinear mixed selectivity supports reliable neural computation. *PLoS Computational Biology, 16*(2), e1007544.

Kelly, R. C., Smith, M. A., Samonds, J. M., Kohn, A., Bonds, A., Movshon, J. A., & Lee, T. S. (2007). Comparison of recordings from microelectrode arrays and single electrodes in the visual cortex. *Journal of Neuroscience, 27*(2), 261-264.

Knierim, J. J., & Van Essen, D. C. (1992). Neuronal responses to static texture patterns in area V1 of the alert macaque monkey. *Journal of Neurophysiology, 67*(4), 961-980.

Kusunoki, M., Moutoussis, K., & Zeki, S. (2006). Effect of background colors on the tuning of color-selective cells in monkey area V4. *Journal of Neurophysiology, 95*(5), 3047-3059.

Li, M., Liu, F., Juusola, M., & Tang, S. (2014). Perceptual color map in macaque visual area V4. *Journal of Neuroscience, 34*(1), 202-217.

Logothetis, N. K., & Sheinberg, D. L. (1996). Visual object recognition. *Annual Review of Neuroscience, 19*(1), 577-621.

Mazer, J. A., & Gallant, J. L. (2003). Goal-related activity in V4 during free viewing visual search: Evidence for a ventral stream visual salience map. *Neuron, 40*(6), 1241-1250.

McClure Jr, J. P., & Polack, P.-O. (2019). Pure tones modulate the representation of orientation and direction in the primary visual cortex. *Journal of Neurophysiology, 121*(6), 2202-2214.

McIntosh, L., Maheswaranathan, N., Nayebi, A., Ganguli, S., & Baccus, S. (2016b). *Deep learning models of the retinal response to natural scenes.* Paper presented at the Advances in Neural Information Processing Systems.

McMahon, D. B., Jones, A. P., Bondar, I. V., & Leopold, D. A. (2014). Face-selective neurons maintain consistent visual responses across months. *Proceedings of the National Academy of Sciences, 111*(22), 8251-8256.

Mirabella, G., Bertini, G., Samengo, I., Kilavik, B. E., Frilli, D., Della Libera, C., & Chelazzi, L. (2007). Neurons in area V4 of the macaque translate attended visual features into behaviorally relevant categories. *Neuron, 54*(2), 303-318.

Morcos, A. S., Barrett, D. G., Rabinowitz, N. C., & Botvinick, M. (2018). On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*.

Motter, B. C. (1994). Neural correlates of attentive selection for color or luminance in extrastriate area V4. *Journal of Neuroscience, 14*(4), 2178-2189.

Okazawa, G., Tajima, S., & Komatsu, H. (2015). Image statistics underlying natural texture selectivity of neurons in macaque V4. *Proceedings of the National Academy of Sciences, 112*(4), E351-E360.

Olshausen, B. A., & Field, D. J. (2006). What is the other 85 percent of V1 doing. *L. van Hemmen, & T. Sejnowski (Eds.), 23*, 182-211.

Osorio, D., & Vorobyev, M. (2008). A review of the evolution of animal colour vision and visual communication signals. *Vision research, 48*(20), 2042-2051.

Pasupathy, A., & Connor, C. E. (2001). Shape representation in area V4: position-specific tuning for boundary conformation. *Journal of Neurophysiology, 86*(5), 2505-2519.

Ponce, C. R., Xiao, W., Schade, P., Hartmann, T. S., Kreiman, G., & Livingstone, M. S. (2019). Evolving super stimuli for real neurons using deep generative networks. *bioRxiv*, 516484.

Portilla, J., & Simoncelli, E. P. (2000). A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision, 40*(1), 49-70.

Pospisil, D. A., Pasupathy, A., & Bair, W. (2018). 'Artiphysiology'reveals V4-like shape tuning in a deep network trained for image classification. *ELife, 7*, e38242.

Rigotti, M., Barak, O., Warden, M. R., Wang, X.-J., Daw, N. D., Miller, E. K., & Fusi, S. (2013). The importance of mixed selectivity in complex cognitive tasks. *Nature, 497*(7451), 585.

Roe, A. W., Chelazzi, L., Connor, C. E., Conway, B. R., Fujita, I., Gallant, J. L., . . . Vanduffel, W. (2012). Toward a unified theory of visual area V4. *Neuron, 74*(1), 12-29.

Rosenthal, I., Ratnasingam, S., Haile, T., Eastman, S., Fuller-Deets, J., & Conway, B. R. (2018). Color statistics of objects, and color tuning of object cortex in macaque monkey. *Journal of vision, 18*(11), 1-1.

Seung, H. S., & Sompolinsky, H. (1993). Simple models for reading neuronal population codes. *Proceedings of the National Academy of Sciences, 90*(22), 10749-10753.

Sharpee, T. O., Kouh, M., & Reynolds, J. H. (2013). Trade-off between curvature tuning and position invariance in visual area V4. *Proceedings of the National Academy of Sciences, 110*(28), 11618-11623.

Shoham, S., Fellows, M. R., & Normann, R. A. (2003). Robust, automatic spike sorting using mixtures of multivariate t-distributions. *Journal of neuroscience methods, 127*(2), 111-122.

Sillito, A., & Jones, H. (1996). Context-dependent interactions and visual processing in V1. *Journal of Physiology-Paris, 90*(3-4), 205-209.

Simoncelli, E. P., & Olshausen, B. A. (2001). Natural image statistics and neural representation. *Annual Review of Neuroscience, 24*(1), 1193-1216.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Tanigawa, H., Lu, H. D., & Roe, A. W. (2010). Functional organization for color and orientation in macaque V4. *Nature Neuroscience, 13*(12), 1542-1548.

Touryan, J., Felsen, G., & Dan, Y. (2005). Spatial structure of complex cell receptive fields measured with natural images. *Neuron, 45*(5), 781-791.

Vinje, W. E., & Gallant, J. L. (2000). Sparse coding and decorrelation in primary visual cortex during natural vision. *Science, 287*(5456), 1273-1276.

Wang, Z., Stocker, A. A., & Lee, D. D. (2012). Optimal neural tuning curves for arbitrary stimulus distributions: Discrimax, infomax and minimum Lp loss. *Advances in Neural Information Processing Systems (NIPS), 3*, 2168-2176.

Wang, Z., Stocker, A. A., & Lee, D. D. (2013). *Optimal neural population codes for high-dimensional stimulus variables.* Paper presented at the Advances in Neural Information Processing Systems.

Yamins, D. L., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., & DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences, 111*(23), 8619-8624.

Yeh, C.-I., Xing, D., Williams, P. E., & Shapley, R. M. (2009). Stimulus ensemble and cortical layer determine V1 spatial receptive fields. *Proceedings of the National Academy of Sciences, 106*(34), 14652-14657.

Zeki, S. (1980). The representation of colours in the cerebral cortex. *Nature, 284*(5755), 412-418.

Zeki, S. M. (1973). Colour coding in rhesus monkey prestriate cortex. *Brain research, 53*(2), 422-427.

Ziemba, C. M., Freeman, J., Movshon, J. A., & Simoncelli, E. P. (2016). Selectivity and tolerance for visual texture in macaque V2. *Proceedings of the National Academy of Sciences, 113*(22), E3140-E3149.

Supplementary information

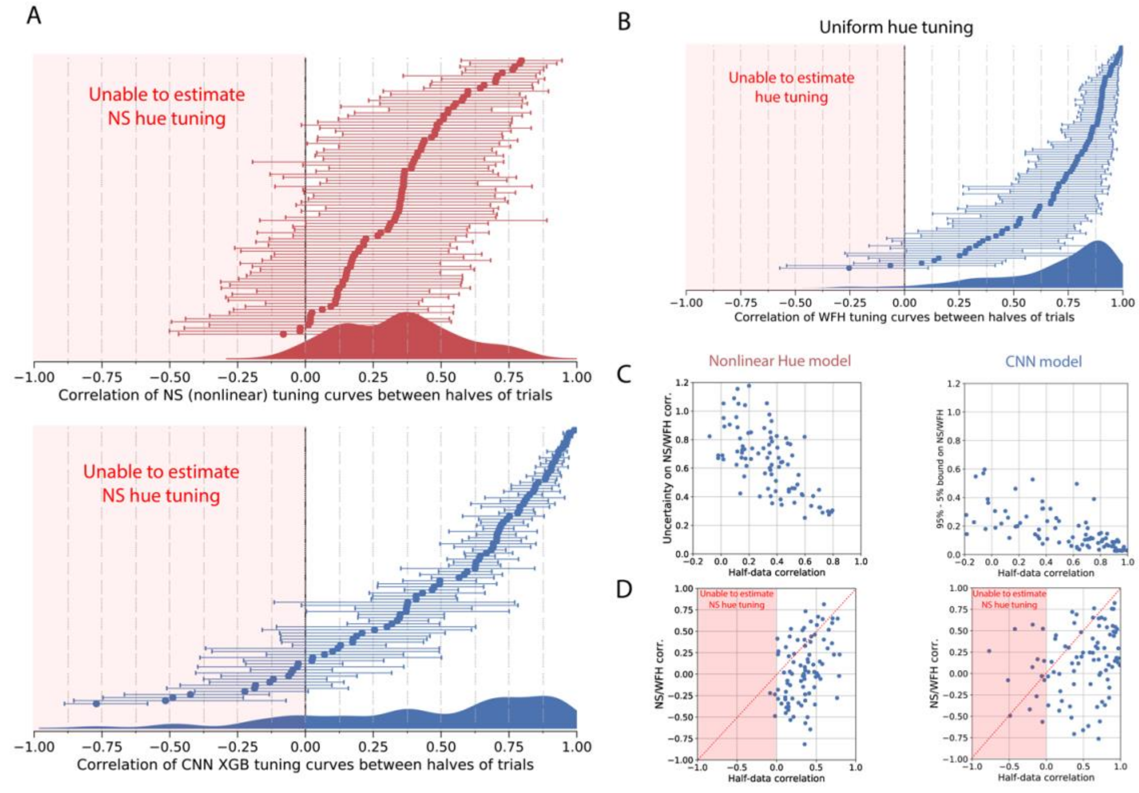


Figure 2-1. Our ability to estimate hue tuning can be captured by the correlation of the tuning estimated on two non-overlapping halves of the trials. This correlation would be 1 in the no-noise or infinite-data condition. For a single neuron, the half-trial correlation represents an estimate of what natural scene/uniform hue tuning curve correlations we would observe if hue tuning did not change between conditions. (Note that by fitting models on only half of the data, the estimate of hue tuning is noisier than in the full-data tuning estimation. Our actual ability to measure hue tuning is thus better than communicated by this control. For this reason these plots show a *lower bound* of the correlation we would expect if hue tuning were the same across conditions.) **A**) For the nonlinear hue model (red) and the CNN model (blue), these plots display the correlation of the tuning estimated on two non-overlapping halves of the trials. One can also consider this test as running 2-fold cross-validation and comparing the tuning curves estimated on both splits of data. Like in the main analysis, we split the data such that all trials (fixations) on the same image were placed in the same fold of data. In this plot the neurons are again ordered by their correlation to produce a cumulative distribution (the order of neurons is not the same as in Figures 2C and 4A). Errors show 5th and 95th percentiles of this procedure repeated on the original data resampled with replacement. The smoothed distributions projected below are reproduced in Figures 2C and 4A. **B)** The half-trial control for the uniform hue condition. This communicates how precisely we can estimate uniform hue tuning. The errors again derive from repeating the cross-half correlation when resampling the trials and re-splitting the data in half. **C)** The estimation error as communicated by these half-data control captures the same sources of variability that were incorporated into the principle uncertainty measure of the correlation between tuning curves (e.g.

Figure 2Bii). That uncertainty was measured by resampling the trials, then re-calculating and re-correlating the tuning curves. To demonstrate this, here we show the relation between the half-data correlation and the size of the uncertainty bars from the main figures (Figures 2C and 4A). As expected, there is a strong negative correlation. Higher half-data correlations for a neuron correspond to smaller bounds of the natural scene/uniform hue correlation. **D)** Here we compare, neuron-by-neuron, the relationship between the half-data correlation and the natural scene/uniform hue correlation. (Panel A only communicates the difference in overall distributions.**)** Importantly, there is little relation between the half-data correlation (i.e. our ability to estimate natural scene hue tuning) and the natural scene/uniform hue correlation (i.e. whether we observed that neuron to shift tuning). This shows when we observe a shift in hue tuning, it is not simply because for that neuron we poorly estimated the natural scene hue tuning. Another key takeaway is the number of neurons that lie in the region below the dotted red line, where the split-trial correlation is higher than the natural scene/uniform hue correlation. For both estimation methods (nonlinear hue and CNN model), significantly more neurons lie below this line than above it.

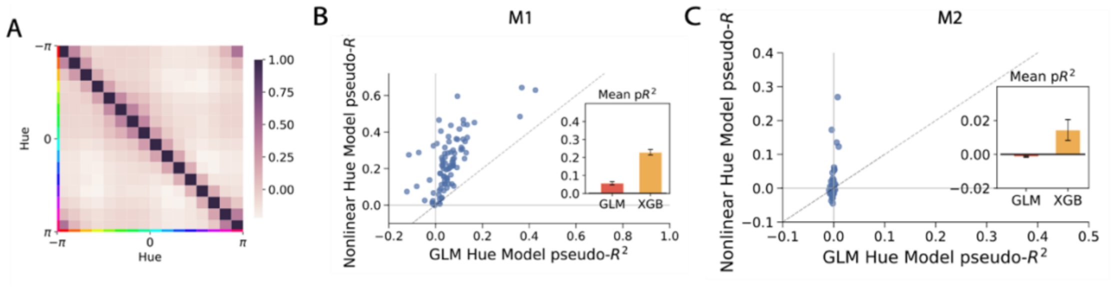


Figure 3-1. The response of V4 neurons to hue is nonlinear and contains interactions between bins of hues. A) The correlation matrix of hues on the natural image dataset observed by M1. Since the off-diagonal terms are not zero, there are correlation between hues (especially of similar colors). These correlations could bias the tuning curve of a linear fit if nonlinear hue interactions exist in the neural response. As shown in (B) and in (C), these interactions do indeed exist. This can be seen by the fact that the nonlinear model (gradient boosted trees, XGB) predicts neural activity better than the generalized linear model (GLM) when both are fed the (saturation-weighted) histograms of hues present within the receptive field during each fixation.

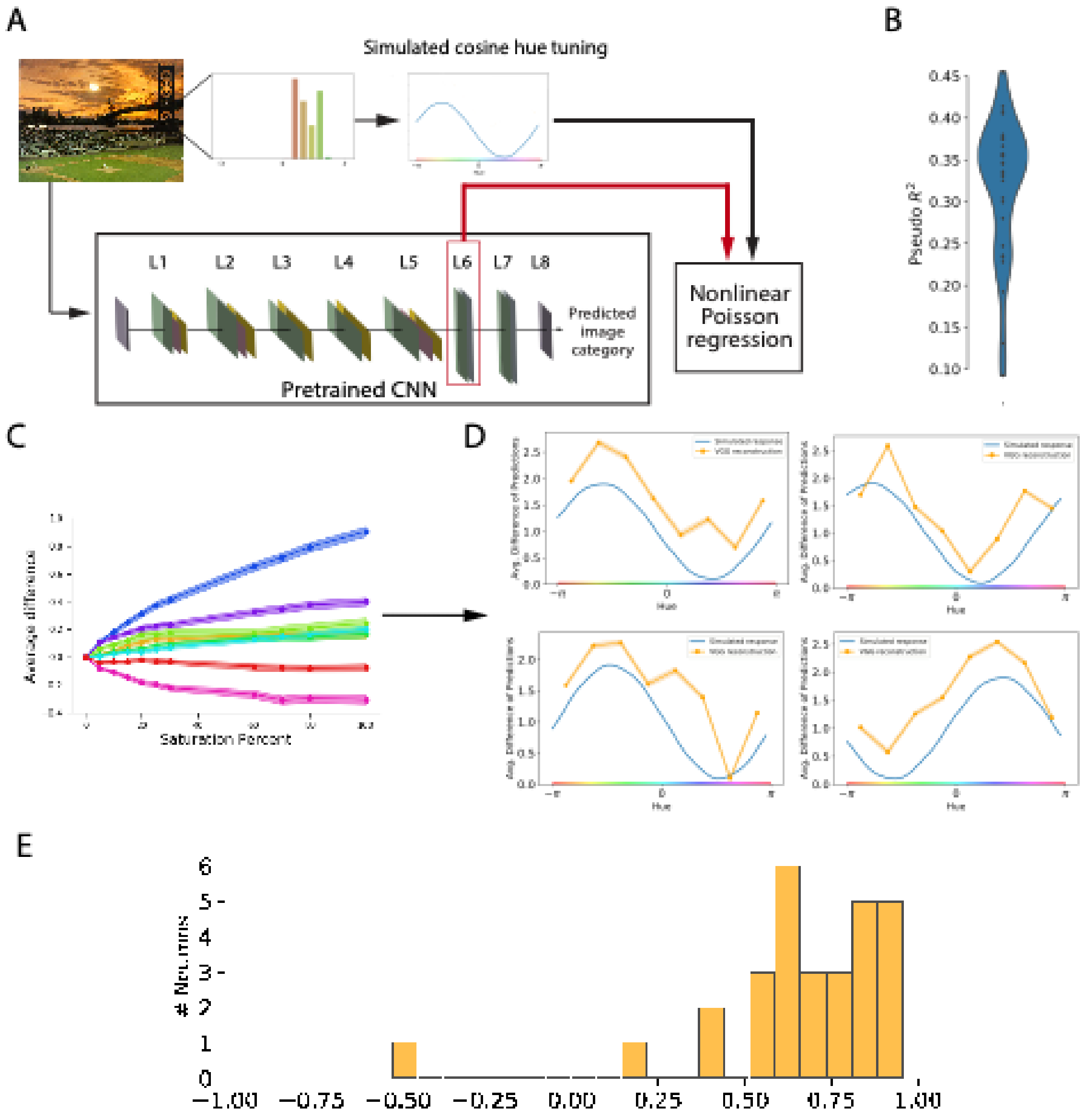


Figure 4-1. Reconstructing simulated neural responses shows that the CNN method can in principle observe hue tuning. A) As for the main model, we fit a nonlinear Poisson regression model to predict 'neural' responses from the intermediate activations of the VGG16 model when given image segments. Instead of the actual neural response, here we fit to a simulated neural response, which comprised of a random (fixed) cosine filter applied to the distribution of hues in each fixation. B) The model could predict these simple responses well, but not perfectly, with typical pseudo-R^2 scores less than 0.4. C) Once again we calculated the average difference in predictions between held-out images and those same images but with each of 8 bins of hues desaturated by

some percentage. We plot the difference in response as a function of desaturation. It can be seen that the line is somewhat sub-linear, like for actual neural responses. This plot proves that some of this sub-linearity is not neural in origin, but rather a function of both our choice of color space (CIELUV) and the way that the VGG model incorporates color into the response. D) Tuning curves constructed in this way (from the slopes of the saturation dependencies) closely resemble the original filters, with some noise. E) The typical noise of this method's reconstruction of tuning curves can be summarized as a distribution of tuning curve correlations. This distribution is the point of comparison, representing what distribution we would expect if hue tuning were unchanged between categories of stimuli.

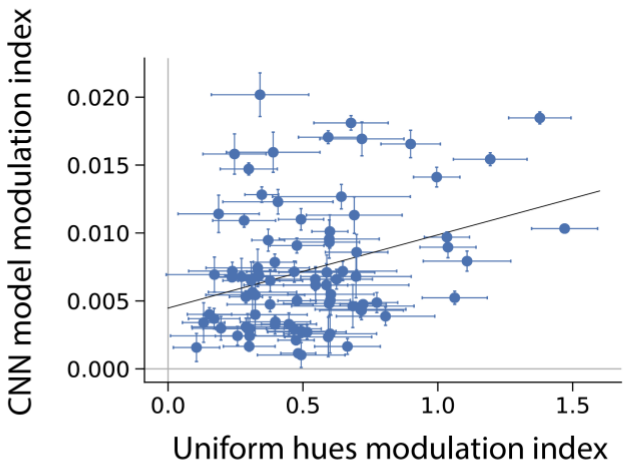


Figure 4-2: We calculated a modulation index measuring how drastically hue affected the V4 response on either uniform hues or natural images. In uniform hues, the modulation index was defined as the maximum of the uniform hue tuning curve, minus the minimum, and divided by the mean spike rate. In natural scenes, we examined how strongly various hues affected the CNN model response. This was measured by the difference between the maximum and the minimum of the CNN model tuning curve, which, measuring a difference in the predictions rather than the absolute value, is already mean-normalized. There was a weak correlation (p=0.003) between these two indices.

Chapter 3: Modern machine learning as benchmark for encoding models

Foreword

This chapter provides a machine learning tool to verify the assumptions of simpler, interpretable encoding models. By acting as performance benchmarks, machine learning methods can identify if nonlinearity remains to be explained. This work is also an instructive demonstration of which machine learning methods are most successful at neural prediction. This paper is an example of **Role 3:** machine learning as a benchmark for simpler models.

In a collaboration with Prof. Lee Miller at Northwestern, we applied this approach to datasets collected from motor and somatosensory cortices. Many of these machine learning methods had not previously been used to predict neural activity, and we found they empirically worked quite well and outperformed GLMs at predicting spikes in three separate brain areas. This result is a warning that GLMs may generally fail to capture neural nonlinearity and can mischaracterize stimulus/response relationships for naïvely chosen sets of features.

This chapter is reproduced from a paper now published in *Frontiers in Computational Neuroscience*[3], for which it was selected as Editor's Pick 2021. It was also presented at two conferences, in the form of a poster at COSYNE 2017 and a talk at SAND8 in 2018.

[3] Benjamin, Ari S., Hugo L. Fernandes, Tucker Tomlinson, Pavan Ramkumar, Chris VerSteeg, Raeed H. Chowdhury, Lee E. Miller, and Konrad P. Kording. "Modern Machine Learning as a Benchmark for Fitting Neural Responses." *Frontiers in computational neuroscience* 12 (2018)

Abstract

Neuroscience has long focused on finding encoding models that effectively ask “what predicts neural spiking?” and generalized linear models (GLMs) are a typical approach. It is often unknown how much of explainable neural activity is captured, or missed, when fitting a model. Here we compared the predictive performance of simple models to three leading machine learning methods: feedforward neural networks, gradient boosted trees (using XGBoost), and stacked ensembles that combine the predictions of several methods. We predicted spike counts in macaque motor (M1) and somatosensory (S1) cortices from standard representations of reaching kinematics, and in rat hippocampal cells from open field location and orientation. Of these methods, XGBoost and the ensemble consistently produced more accurate spike rate predictions and were less sensitive to the preprocessing of features. These methods can thus be applied quickly to detect if feature sets relate to neural activity in a manner not captured by simpler methods. Encoding models built with a machine learning approach accurately predict spike rates and can offer meaningful benchmarks for simpler models.

Introduction

A central tool of neuroscience is the tuning curve, which maps aspects of external stimuli to neural responses. The tuning curve can be used to determine what information a neuron encodes in its spikes. For a tuning curve to be meaningful it is important that it accurately describes the neural response. Often, however, methods are chosen for simplicity but not evaluated for their relative accuracy. Since inaccurate methods may systematically miss aspects of the neural response, any choice of predictive method should be compared with accurate benchmark methods.

A popular predictive model for neural data is the Generalized Linear Model (GLM) (Gerwinn, Macke, & Bethge, 2010; Nelder & Baker, 1972; Simoncelli, Paninski, Pillow, & Schwartz, 2004; Truccolo, Eden, Fellows, Donoghue, & Brown, 2005; Wu, David, &

Gallant, 2006). The GLM performs a nonlinear operation upon a linear combination of the input features, which are often called external covariates. Typical covariates are stimulus features, movement vectors, or the animal's location, and may include covariate history or spike history. In the absence of history terms, the GLM is also referred to as a linear-nonlinear Poisson (LN) cascade. The nonlinear operation is usually held fixed, though it can be learned (Chichilnisky, 2001; Paninski, Fellows, Hatsopoulos, & Donoghue, 2004), and the linear weights of the combined inputs are chosen to maximize the agreement between the model fit and the neural recordings. This optimization problem of weight selection is convex, allowing a global optimum, and can be solved with efficient algorithms (Paninski, 2004). The assumption of Poisson firing statistics can often be loosened (Pillow, Paninski, Uzzell, Simoncelli, & Chichilnisky, 2005), as well, allowing the modeling of a broad range of neural responses. Due to its ease of use, perceived interpretability, and flexibility, the GLM has become a popular model of neural spiking.

When using a GLM, it is important to check that the method's assumptions about the data are correct. The GLM's central assumption is that the inputs relate linearly to the log firing rate, or generally some monotonic function of the firing rate. It thus cannot learn arbitrary multi-dimensional functions of the inputs. When the nonlinearity is different than assumed, it is likely that the optimal weight on one input will depend on the values of other inputs. In this case the GLM will only partially represent the neural response, will poorly predict activity, and may not be reproducible on other datasets. This drawback has been noted before, and indeed the GLM has been shown to miss nonlinearity in numerous circumstances (Butts, Weng, Jin, Alonso, & Paninski; Freeman et al.; Heitman et al.; McIntosh, Maheswaranathan, Nayebi, Ganguli, & Baccus). However, GLMs are still commonly applied without comparison to other methods. To test if the linearity assumption is valid, it is sufficient to test if other nonlinear methods predict activity more accurately from the same features. Many extensions have been proposed that introduce a specific form of nonlinearity (Latimer, Chichilnisky, Rieke, & Pillow; Maheswaranathan, Baccus, & Ganguli; McFarland, Cui, & Butts; Theis, Chagas, Arnstein, Schwarz, & Bethge; Williamson, Sahani, & Pillow), but these methods ask specific research questions and are not intended as general benchmarks. What is needed is are nonlinear methods that are universally applicable to new data.

Machine learning (ML) methods for regression have improved dramatically since the invention of the GLM. Many ML methods require little feature engineering (i.e. pre-transformations the features) and do not need to assume linearity. These methods are thus ideal candidates for benchmark methods. The ML approach is now quite standardized and robust across many domains of data. As exemplified by winning solutions on Kaggle, an ML competition website ("Kaggle Winner's Blog," 2016), the usual approach is to fit several top performing methods, and then to ensemble these models together. These methods are now relatively easy to implement in a few lines of code in a scripting language such as Python, and are enabled by well-supported machine learning packages, such as scikit-learn (Pedregosa et al., 2011), Keras (Chollet, 2015), Tensorflow (Abadi et al., 2016), and XGBoost (Chen & Guestrin, 2016a). The greatly increased predictive power of modern ML methods is now very accessible and could help to benchmark and improve the state of the art in encoding models across neuroscience.

In order to investigate the feasibility of ML as a benchmark approach, we applied several ML methods, including artificial neural networks, gradient boosted trees, and ensembles to the task of predicting spike rates, and evaluated their performance alongside a GLM. We compared the methods on data from three separate brain areas. These areas differed greatly in the effect size of covariates and in their typical spike rates, and thus served to evaluate the strengths of these methods across different conditions. In each area we found that the ensemble of methods could more accurately predict spiking than the GLM with typical feature choices. The use of an ML benchmark thus made clear that tuning curves built for these features with a GLM would not capture the full nature of neural activity. We provide our implementing code at https://github.com/KordingLab/spykesML so that all neuroscientists may easily test and compare ML to their own methods on other datasets.

Materials and Methods

Data

We tested our methods at predicting spike rates for neurons in the macaque primary motor cortex, the macaque primary somatosensory cortex, and the rat hippocampus. All animal use procedures were approved by the institutional animal care and use committees at Northwestern University and conform to the principles outlined in the Guide for the Care and Use of Laboratory Animals (National Institutes of Health publication no. 86-23, revised 1985). Data presented here were previously recorded for use with multiple analyses. Procedures were designed to minimize animal suffering and reduce the number used.

The macaque motor cortex data consisted of previously published electrophysiological recordings from 82 neurons in the primary motor cortex (M1) (Stevenson et al.). The neurons were sorted from recordings made during a two-dimensional center-out reaching task with eight targets. In this task the monkey grasped the handle of a planar manipulandum that controlled a cursor on a computer screen and simultaneously measured the hand location and velocity (Fig. 1). After training, an electrode array was implanted in the arm area of area 4 on the precentral gyrus. Spikes were discriminated using offline sorter (Plexon, Inc), counted and collected in 50-ms bins. The neural recordings used here were taken in a single session lasting around 13 minutes.

The macaque primary somatosensory cortex (S1) data was recorded during a two-dimensional random-pursuit reaching task and was previously unpublished. In this task, the monkey gripped the handle of the same manipulandum. The monkey was rewarded for bringing the cursor to a series of randomly positioned targets appearing on the screen. After training, an electrode array was implanted in the arm area of area 2 on the postcentral gyrus, which receives a mix of cutaneous and proprioceptive afferents. Spikes were processed as for M1. The data used for this publication derives from a single recording session lasting 51 minutes.

As with M1 (described in results), we processed the hand position, velocity, and acceleration accompanying the S1 recordings in an attempt to obtain linearized features. The features $(x, y, \dot{x}, \dot{y})$ were found to be the most successful for the GLM. Since cells in the arm area of S1 have been shown to have approximately sinusoidal tuning curves relating to movement direction (Prud'homme & Kalaska, 1994), we also tested the same feature transformations as were performed for M1 but did not observe any increase in predictive power.

The third dataset consists of recordings from 58 neurons in the CA1 region of the rat dorsal hippocampus during a single 93 minute free foraging experiment, previously published and made available online by the authors (Mizuseki, Sirota, Pastalkova, & Buzsáki, 2009a, 2009b). Position data from two head-mounted LEDs provided position and heading direction inputs. Here we binned inputs and spikes from this experiment into 50ms bins. Since many neurons in the dorsal hippocampus are responsive to the location of the rat, we processed the 2D position data into a list of squared distances from a 5x5 grid of place fields that tile the workspace. Each position feature thus has the form

$$p_{ij} = \frac{1}{2}\left(x(t) - \mu_{ij}\right)^T \Sigma_{ij}^{-1}\left(x(t) - \mu_{ij}\right),$$

where μ_{ij} is the center of place field $i, j \leq 5$ and Σ_{ij} is a covariance matrix chosen for the uniformity of tiling. An exponentiated linear combination of the p_{ij} (as is performed in the GLM) evaluates to a single Gaussian centered anywhere between the place fields. The inclusion of the p_{ij} as features thus transforms the standard representation of cell-specific place fields (Brown, Frank, Tang, Quirk, & Wilson, 1998) into the mathematical formulation of a GLM. The final set of features included the p_{ij} as well as the rat speed and head orientation.

Treatment of Spike and Covariate History

We slightly modified our data preparation methods for spike rate prediction when spike and covariate history terms were included as regressors (Fig. 6). To construct spike and covariate history filters, we convolved 10 raised cosine bases (built as in (Pillow et al., 2008)) with binned spikes and covariates. The longest temporal basis included times up to 250 ms before the time bin being predicted. This process resulted in 120 total covariates per sample (10 current covariates, 100 covariate temporal filters, and 10 spike history filters). We predicted spike rates in 5 ms bins (rather than 50 ms) to allow for modeling of more precise time-dependent phenomena, such as refractory effects. The cross-validation scheme also differs from the main analysis of this paper, as using randomly selected splits of the data would result in the appearance in the test set of samples that were in history terms of training sets, potentially resulting in overfitting. We thus employed a cross-validation routine to split the data continuously in time, assuring that no test set sample has appeared in any form in training sets.

Generalized Linear Model

The Poisson generalized linear model is a multivariate regression model that describes the instantaneous firing rate as a nonlinear function of a linear combination of input features (see e.g. (Aljadeff et al., 2016; Schwartz, Pillow, Rust, & Simoncelli, 2006) for review, (Fernandes, Stevenson, Phillips, Segraves, & Kording, 2014; Pillow et al., 2008; Ramkumar et al., 2016) for usage). Here, we took the form of the nonlinearity to be exponential, as is common in previous applications of GLMs to similar data (Saleh, Takahashi, & Hatsopoulos, 2012). It should be noted that it is also possible to learn arbitrary link functions through histogram methods (Chichilnisky, 2001; Paninski, Fellows, et al., 2004). We approximate neural activity as a Poisson process, in which the probability of firing in any instant is independent of firing history. The general form of the GLM is depicted in Figure 1. We implemented the GLM using elastic-net regularization, using the open-source Python package pyglmnet (Ramkumar et al., 2017). The regularization path was optimized separately on a single neuron in each dataset on a validation set not used for scoring.

Neural Network

Neural networks are well-known for their success at supervised learning tasks. More comprehensive reviews can be found elsewhere (Schmidhuber, 2015). Here, we implemented a simple feedforward neural network and, for the analysis with history terms, an LSTM, a recurrent neural network architecture that allows the modeling of time dependencies on multiple time-scales (Gers, Schmidhuber, & Cummins, 2000).

We point out that a feedforward neural network with no hidden layers is equivalent in mathematical form to a GLM (Fig. 1). For multilayer networks, one can write each hidden layer of n nodes as simply n GLMs, each taking the output of the previous layer as inputs (noting that the weights of each are chosen to maximize only the final objective function, and that the intermediate nonlinearities need not be the same as the output nonlinearity).

A feedforward neural network can be seen as a generalization, or repeated application of a GLM.

The networks were implemented with the open-source neural network library Keras, running Theano as the backend (Chollet, 2015; Team et al., 2016). The feedforward network contained two hidden layers, dense connections, rectified linear activation, and a final exponentiation. To help avoid overfitting, we allowed dropout on the first layer, included batch normalization, and allowed elastic-net regularization upon the weights (but not the bias term) of the network (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov). The networks were trained to maximize the Poisson likelihood of the neural response. We optimized over the number of nodes in the first and second hidden layers, the dropout rate, and the regularization parameters for the feedforward neural network, and for the number of epochs, units, dropout rate, and batch size for the LSTM. Optimization was performed on only a subset of the data from a single neuron in each dataset, using Bayesian optimization (Snoek, Larochelle, & Adams) in an open-source Python implementation ("BayesianOptimization," 2016).

Gradient Boosted Trees

A popular method in many machine learning competitions is that of gradient boosted trees. Here we describe the general operation of XGBoost, an open-source implementation that is efficient and highly scalable, works on sparse data, and easy to implement out-of-the-box (Chen & Guestrin, 2016a).

XGBoost trains many sequential models to minimize the residual error of the sum of previous model. Each model is a decision tree, or more specifically a classification and regression tree (CART) (J. H. Friedman, 2001). Training a decision tree amounts to determining a series of rule-based splits on the input to classify output. The CART algorithm generalizes this to regression by taking continuously-valued weights on each of the leaves of the decision tree.

For any predictive model $\hat{y}^{(1)} = f_1(\boldsymbol{x_i})$ and true response y_i, we can define a loss function $l\big(\hat{y}^{(1)}, y_i\big)$ between the prediction and the response. The objective to be minimized during training is then simply the sum of the loss over each training example i, plus some regularizing function Ω that biases towards simple models.

$$L = \sum_i l(\hat{y}_i^{(1)}, y_i) + \Omega(f_1)$$

After minimizing L for a single tree, XGBoost constructs a second tree $f_2(\boldsymbol{x_i})$ that approximates the residual. The objective to be minimized is thus the total loss L between the true response y_i and the sum of the predictions given by the first tree and the one to be trained.

$$L = \sum_i l(\hat{y}_i^{(1)} + f_2(\boldsymbol{x}_i), y_i) + \Omega(f_2)$$

This process is continued sequentially for a predetermined number of trees, each trained to approximate the residual of the sum of previous trees. In this manner XGBoost is designed to progressively decrease the total loss with each additional tree. At the end of training, new predictions are given by the sum of the outputs of all trees.

$$\hat{y} = \sum_{k=1}^{N} f_k(\boldsymbol{x})$$

In practice, it is simpler to choose the functions f_k via gradient boosting, which minimizes a second order approximation of the loss function (J. Friedman, Hastie, & Tibshirani, 2000).

XGBoost offers several additional parameters to optimize performance and prevent overfitting. Many of these describe the training criteria for each tree. We optimized some of these parameters for a single neuron in each dataset using Bayesian optimization (again over a validation set different from the final test set). These parameters included the number of trees to train, the maximum depth of each decision tree, and the minimum weight allowed on each decision leaf, the data subsampling ratio, and the minimum gain required to create a new decision branch.

Random Forests

We implement random forests here to increase the power of the ensemble (see below); their performance alone is displayed in Supplementary Figure 1. It should be noted that the Scikit-learn implementation currently only minimizes the mean-squared error of the output, which is not properly applicable to Poisson processes and may cause poor performance. Despite this drawback their presence still improves the ensemble scores. Random forests train multiple parallel decision trees on the features-to-spikes regression problem (not sequentially on the remaining residual, as in XGBoost) and averages their outputs (Ho, 1998). The variance on each decision tree is increased by training on a sample of the data drawn with replacement (i.e., bootstrapped inputs) and by choosing new splits using only a random subset of the available features. Random forests are implemented in Scikit-learn (Pedregosa et al., 2011).

Ensemble Method

It is a common machine learning practice to create ensembles of several trained models. Different algorithms may learn different characteristics of the data, make different types of errors, or generalize differently to new examples. Ensemble methods allow for the successes of different algorithms to be combined. Here we implemented *stacking*, in which the output of several models is taken as the input set of a new model (David H Wolpert, 1992). After training the GLM, neural network, random forest, and XGBoost on the features of each dataset, we trained an additional instance of XGBoost using the spike rate predictions of the previous methods as input. The outputs of this 'second stage' XGBoost are the predictions of the ensemble.

Scoring and Cross-Validation

Each of the three methods was scored with the Poisson pseudo-R2 score, a scoring function applicable to Poisson processes (Cameron & Windmeijer, 1997). Note that a standard R2 score assumes Gaussian noise and cannot be applied here. The pseudo-R2 was calculated as one minus the ratio of the deviances of the predicted output $\hat{y}$ to the mean firing rate $\bar{y}$.

$$R_M^2 = 1 - \frac{D(\hat{y})}{D(\bar{y})}$$

We can gain intuition into the pseudo-R2 score by writing out the deviances in terms of log likelihoods $L()$, and combining the fraction.

$$R_M^2 = 1 - \frac{\log L(y) - \log L(\hat{y})}{\log L(y) - \log L(\bar{y})} = \frac{\log L(\hat{y}) - \log L(\bar{y})}{\log L(y) - \log L(\bar{y})}$$

This expression includes $L(y)$, which is the log likelihood of the "saturated model", which offers one parameter per observation and models the data perfectly. The pseudo-R2 can thus be interpreted as the fraction of the maximum potential log-likelihood gain achieved by the tested model (Cameron & Windmeijer, 1997). It takes a value of 0 when the data is as likely under the tested model as the null model, and a value of 1 when the tested model perfectly describes the data. It is empirically a lower value than a standard R2 when both are applicable (Domencich & McFadden, 1975). The null model can also be taken to be a model other than the mean firing rate (e.g. the GLM) to directly compare two methods, in which case we refer to the score as the 'comparative pseudo-R2'. The comparative pseudo-R2 is referred to elsewhere as the 'relative pseudo-R2', renamed here to avoid confusion with the difference of two standard pseudo-R2 scores both measured against the mean (Fernandes et al., 2014).

We used 8-fold cross-validation (CV) when assigning a final score to the models. The input and spike data were segmented into eight equal partitions. These partitions were continuous in time when spike and covariate history were included as covariates, and otherwise were segmented randomly in time. The methods were trained on seven partitions and tested on the eighth, and this was repeated until all segments served as the test partition once. The mean of the eight scores are then recorded for the final score.

Cross-validation for ensemble methods requires extra care since the inputs for the ensemble are themselves model predictions for each data point. The training set for the ensemble must contain predictions from methods that were themselves not trained on the validation set. Otherwise, there may be a leak of information from the validation set into the training set and the validation score might be better than on a true held-out set. This rules out using simple k-fold CV with all methods and the ensemble trained on the same test/train splits. Instead, we used a nested CV scheme to train and score the ensemble. We create an outer j=8 folds to build training and test sets for the ensemble. On each outer fold we create first-order predictions for each data point in the following manner. We first run an inner k-fold CV on just the training set (i.e. 7/8 of the original dataset) with each first stage method such that we obtain predictions for the whole training set of that fold. This ensures that the ensemble's test set was never used for training any method. Finally, we build the ensemble's test set from the predictions of the first stage methods trained on the entire training set. The ensemble can then be tested on a held-out set that was never used to fit any model. The process is repeated for each of the j folds and the mean and variance of the j scores of the ensemble's predictions are recorded.

Results

We applied several machine learning methods to predict spike counts in three brain regions and compared the quality of the predictions to those of a GLM. Our primary analysis centered on neural recordings from the macaque primary motor cortex (M1) during reaching (Fig. 1). We examined the methods' relative performance on several sets of movement features with various levels of preprocessing, including one set that included spike and covariate history terms. Analyses of data from rhesus macaque S1 and rat hippocampus indicate how these methods compare for areas other than M1. On each of the three datasets we trained a GLM and compared it to the performance of a feedforward neural network, XGBoost (a gradient boosted trees implementation), and an ensemble method. The ensemble was an additional instance of XGBoost trained on the predictions of all three methods plus a random forest regressor. The application of these methods allowed us to demonstrate the potential of a modern approach to be able to identify whether there are typically neural nonlinearities that are not captured by a GLM. The code implementing these methods can be used by any electrophysiology lab to benchmark their own encoding models.

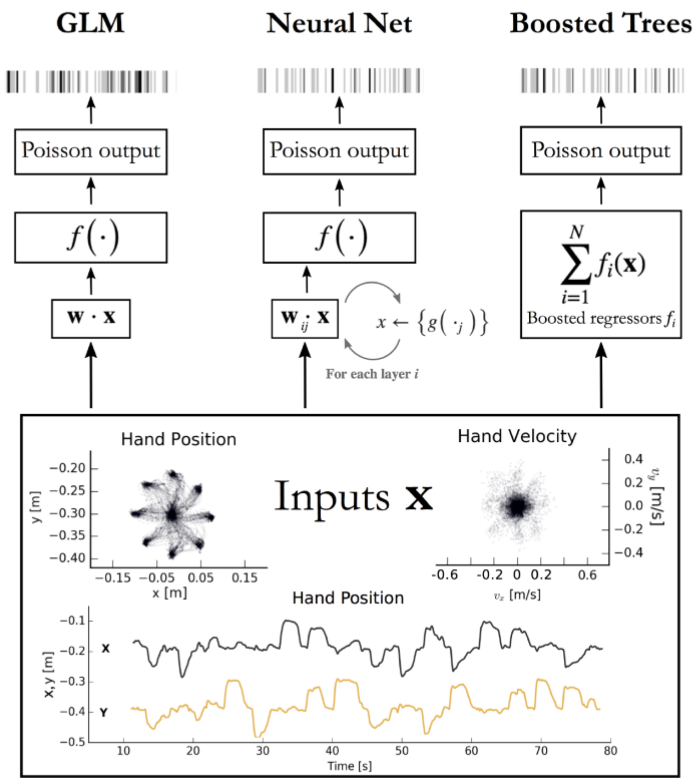


Figure 3-1: Encoding models aim to predict spikes, top, from input data, bottom. The inputs displayed are the position and velocity signals from the M1 dataset **(Stevenson et al., 2011)** but could represent any set of external covariates. The GLM takes a linear combination of the inputs, applies an exponential function *f*, and produces a Poisson spike probability that can be used to generate spikes (left). The feedforward neural network (center) does the same when the number of hidden layers *i* = 0. With *i* ≥ 1 hidden layers, the process repeats; each of the *j* nodes in layer *i* computes a nonlinear function *g* of a linear combination of the previous layer. The vector of outputs from all *j* nodes is then fed as input to the nodes in the next layer, or to the final exponential *f* on the final iteration. Boosted trees (right) return the sum of N functions of the original inputs. Each of the f_i is built to minimize the residual error of the sum of the previous $f_{0:i-1}$.

To test that all methods work reasonably well in a trivial case, we trained each to predict spiking from a simple, well-understood feature. Some neurons in M1 have been described as responding linearly to the exponentiated cosine of movement direction relative to a preferred angle (Amirikian & Georgopulos, 2000). We therefore predicted the spiking of M1 neurons from the cosine and sine of the direction of hand movement in the reaching task. (The linear combination of a sine and cosine curve is a phase-shifted cosine, by identity, allowing the GLM to learn the proper preferred direction). We observed that each method identified a similar tuning curve (Fig. 2b) and that the bulk of the neurons in the dataset were just as well predicted by each of the methods (Fig. 2a, c) (though the ensemble was slightly more accurate than the GLM, with mean comparative pseudo-R^2 above zero, 0.06 [0.043 – 0.084], 95% bootstrapped confidence interval (CI)). The similar performance suggested that, for the majority of neurons, an exponentiated cosine successfully approximates the response to movement direction alone, as has been previously found (Paninski, Shoham, Fellows, Hatsopoulos, & Donoghue, 2004). All methods can in principle estimate tuning curves, and machine learning can indicate if the proper features are used.

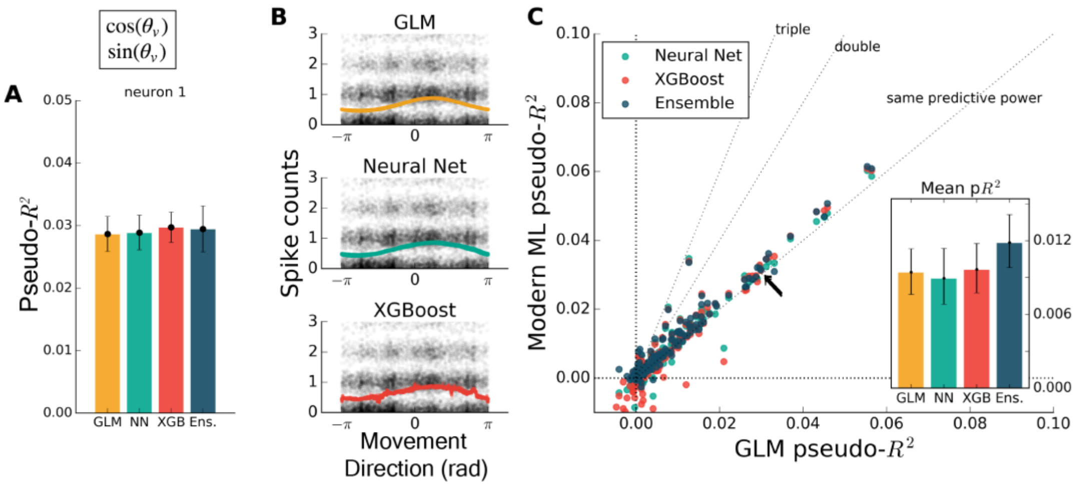


Figure 2: **Encoding models of M1 performed similarly when trained on the sine and cosine of hand velocity direction**. All methods can in principle estimate tuning curves. (a) The pseudo-R^2 for an example neuron was similar for all four methods. On this figure and in Figures 3-5 the example neuron is the same, and is not the neuron for which method hyperparameters were optimized. (b) We constructed tuning curves by plotting the predictions of spike rate on the validation set against movement direction. The black points are the recorded responses, to which we added y-axis jitter for visualization to better show trends in the naturally quantized levels of binned spikes. The tuning curves of the neural net and XGBoost were similar to that of the GLM. The tuning curve of the ensemble method was similar and is not shown. (c) Plotting the pseudo-R^2 of modern ML methods vs. that of the GLM indicates that the similarity of methods generalizes across neurons. The single neuron plotted at left is marked with black arrows. The mean scores, inset, indicate the overall success of the methods; error bars represent the 95% bootstrap confidence interval.

If the form of the nonlinearity is not known, machine learning can still attain good predictive ability. To illustrate the ability of modern machine learning to find the proper nonlinearity, we performed the same analysis as above but omitted the initial cosine feature-engineering step. Trained on only the hand velocity direction, in radians, which changes discontinuously at ±π, all methods but the GLM closely matched the predictive power they attained using the engineered feature (Fig. 3a). The GLM failed at generating a meaningful tuning curve, which was expected since the exponentiated velocity direction is not equal to cosine tuning (Fig. 3b). Both trends were consistent across the population of recorded neurons (Fig. 3c). The neural net, XGBoost, and ensemble methods can learn the nonlinearity of single features without requiring manual feature transformation.

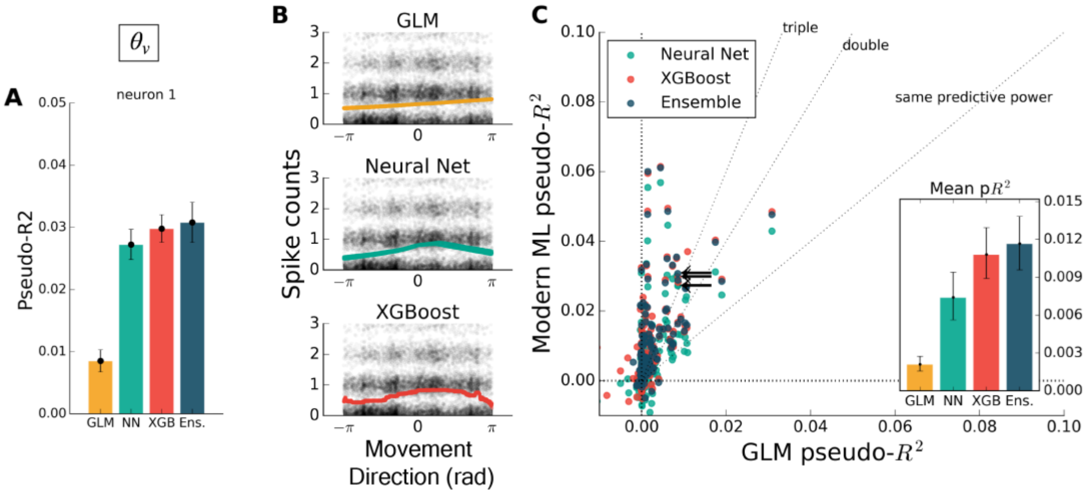


Figure 3: Modern ML models learn the cosine nonlinearity when trained on hand velocity direction, in radians. (a) For the same example neuron as in Figure 2, the neural net and XGBoost maintained the same predictive power, while the GLM was unable to extract a relationship between direction and spike rate. (b) XGBoost and neural nets displayed reasonable tuning curves, while the GLM reduced to the average spiking rate (with a small slope, in this case). (c) Most neurons in the population were poorly fit by the GLM, while the ML methods achieved the performance levels of Figure 2. The ensemble performed the best of the methods tested.

The inclusion of multiple features raises the possibility of nonlinear feature interactions that may elude a GLM. As a simple demonstration of this principle, we trained all methods on the four-dimensional set of hand position and velocity $(x, y, \dot{x}, \dot{y})$. While all methods gained predictive power relative to models using movement direction alone, the GLM failed to match the other methods (Fig. 4a, c). If the GLM was fit alone, and no further featuring engineering been attempted, these features would have appeared to be relatively uninformative of the neural response. If nonlinear interactions exist between preselected features, machine learning methods can potentially learn these interactions and indicate if more linearly-related features exist.

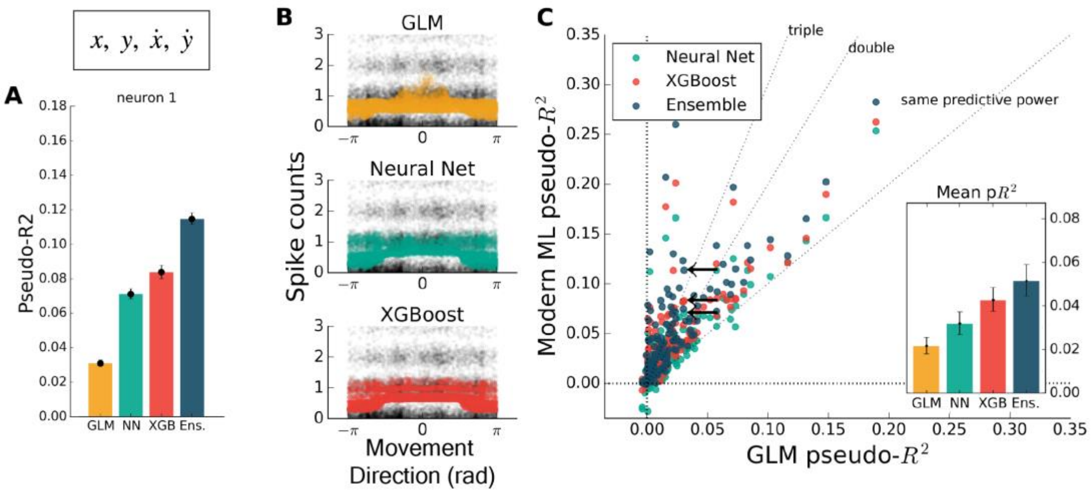


Figure 4: Modern ML methods can learn nonlinear interactions between features. Here the methods are trained on the feature set $(x, y, \dot{x}, \dot{y})$. Note the change in axes scales from Figures 2-3. (a) For the same example neuron as in Figure 3, all methods gained a significant amount of predictive power, indicating a strong encoding of position and speed or their correlates. The GLM showed less predictive power than the other methods on this feature set. (b) The spike rate in black, with jitter on the y-axis, again overlaid with the predictions of the three methods plotted against velocity direction. The projection of the multidimensional tuning curve onto a 1D velocity direction dependence leaves the projected curve diffuse. (c) The ensemble method, neural network, and XGBoost performed consistently better than the GLM across the population. The mean pseudo-R^2 scores show the hierarchy of success across methods.

While feature engineering can improve the performance of GLMs, it is not always simple to guess the optimal set of processed features. We demonstrated this by training all methods on features that have previously been successful at explaining spike rate in a similar center-out reaching task (Paninski, Fellows, et al., 2004). These extra features included the sine and cosine of velocity direction (as in Figure 2), and the speed, radial distance of hand position, and the sine and cosine of position direction. The training set was thus 10-dimensional, though highly redundant, and was aimed at maximizing the predictive power of the GLM. Feature engineering improved the predictive power of all methods to variable degrees, with the GLM improving to the level of the neural network (Fig. 5). XGBoost and the ensemble still predicted spike rates better than the GLM (Fig. 5c), with the ensemble scoring on average nearly double the GLM (ratio of population means of 1.8 [1.4 – 2.2], 95% bootstrapped CI). The ensemble was significantly better than XGBoost (mean comparative pseudo-R^2 of 0.08 [0.055 – 0.103], 95% bootstrapped CI) and was thus consistently the best predictor. Though standard feature engineering greatly improved the GLM, the ensemble and XGBoost still could identify that neural nonlinearity was missed by the GLM.

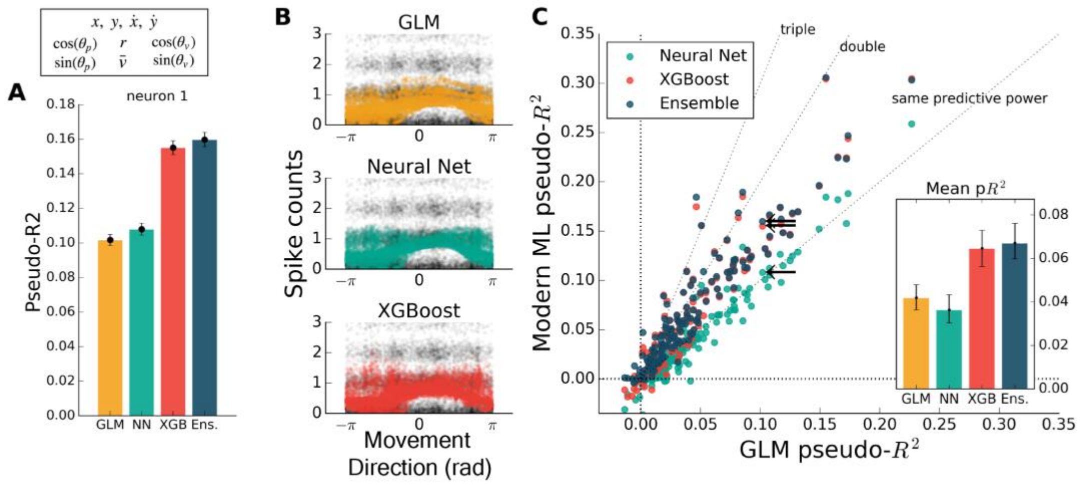


Figure 5: Modern ML methods outperform the GLM with standard featuring engineering. For this figure, all methods were trained on the features $(x, y, \dot{x}, \dot{y})$ plus the engineered features. (a) For this example neuron, inclusion of the computed features increased the predictive power of the GLM to the level of the neural net. All methods increased in predictive power. (b) The tuning curves for the example neuron are diffuse when projected onto the movement direction, indicating a high-dimensional dependence. (c) Even with feature engineering, XGBoost and the ensemble consistently achieve pseudo-R^2 scores higher than the GLM, though the neural net does not. The neuron selected at left is marked with black arrows.

It is important to note that the specific ordering of methods depends on features such as the amount of data available for training. We investigated this dependence for the M1 dataset by plotting the cross-validated performance as a function of the fraction of the data used for training (Supp. Fig. 3). Some neurons are best fit by the GLM when very little data is available, while other neurons are best fit by XGBoost and the ensemble for any amount of data tested. The neural network is most sensitive to training data availability. This sensitivity to the domain of data emphasizes the importance of the applied ML paradigm of evaluating (and potentially ensembling) many methods.

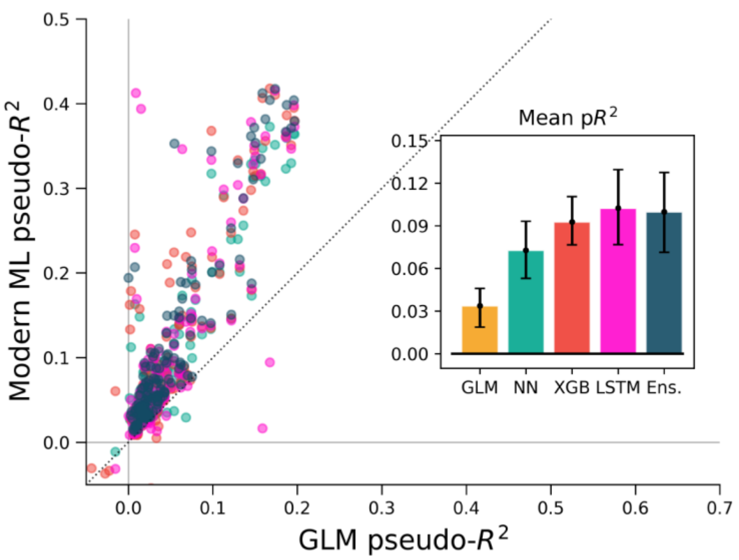


Figure 6: ML algorithms outperform a GLM when covariate history and neuron spike history are included. The feature set of Fig 5 (in macaque M1) was augmented with spike and covariate history terms, so that spike rate was predicted for each 5 ms time bin from the past 250 ms of covariates and neural activity. Cross-validation methods for this figure differ from other figures (see methods) and pseudo-R^2 scores should not be compared directly across figures. All methods outperform the GLM, indicating that the inclusion of history terms does not alone allow the GLM to capture the full nonlinear relationship between covariates and spike rate.

Studies employing a GLM often include activity history as a covariate when predicting spike rates, as well as past values of the covariates themselves, and it is known that this allows GLMs to model a wider range of phenomena (Weber & Pillow, 2016). We tested various ML methods on the M1 dataset using this history-augmented feature set to see if all methods would still explain a similar level of activity. We binned data by 5 ms (rather than 50 ms) to agree in timescale with similar studies, and built temporal filters by convolving 10 raised-cosine bases with features and spikes. We note that smaller time bins result in a sparser dataset, and thus pseudo-R^2 scores cannot be directly compared with other analysis in this paper. On this problem, our selected ML algorithms again outperformed the GLM (Fig. 6). The overall best algorithm was the LSTM, which we include here as it specifically designed for modeling time series, though for most neurons XGBoost performed similarly. Thus, for M1 neurons, the GLM did not capture all predicable phenomena even when spike and covariate history were included.

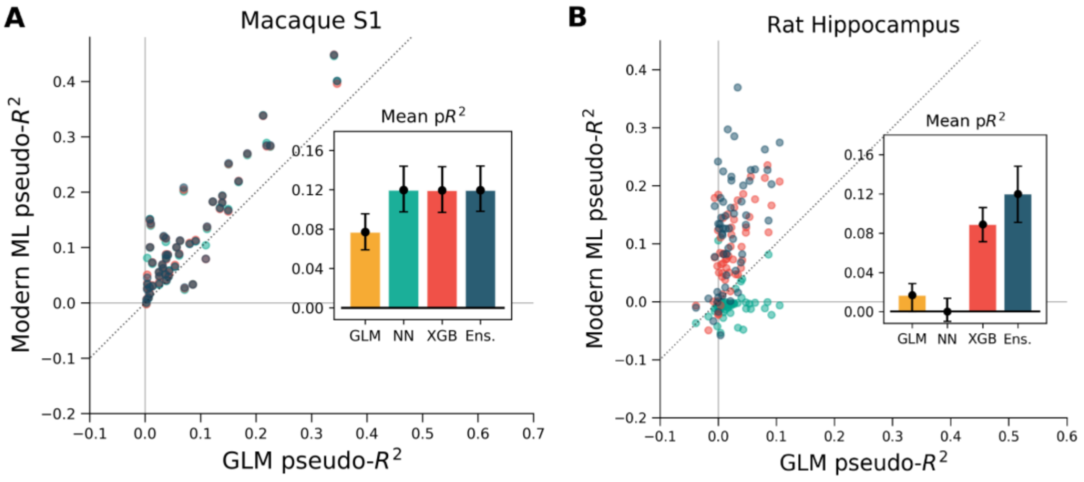


Figure 7: XGBoost and the ensemble method predicted the activity of neurons in S1 and in hippocampus better than a GLM. The diagonal dotted line in both plots is the line of equal predictive power with the GLM. (a) All methods outperform the GLM in the macaque S1 dataset. Interestingly, the neural network, XGBoost and the ensemble scored very similarly for each neuron in the 52 neuron dataset. (b) Many neurons in the rat hippocampus were described well by XGBoost and the ensemble but poorly by the GLM and the neural network. The poor neural network performance in the hippocampus was due to the low rate of firing of most neurons in the dataset (Supp. Fig. 2). Note the difference in axes; hippocampal cells are generally more predictable than those in S1.

To ensure that these results were not specific to the motor cortex, we extended the same analyses to primary somatosensory cortex (S1) data. We again predicted neural activity from hand movement and speed, and here without spike or covariate history terms. The ML methods outperformed the GLM for all but three of the 52 neurons, indicating that firing rates in S1 generally relate nonlinearly to hand position and velocity (Fig. 7a). Each of the three ML methods performed similarly for each neuron. The S1 neural function was thus equally learnable by each method, which is surprising given the dissimilarity of the neural network and XGBoost algorithms. This situation would occur if learning has saturated near ground truth, though this cannot be proven definitively to be the case. It is at least clear from the underperformance of the GLM that the relationship of S1 activity to these covariates is nonlinear beyond the assumptions of the GLM.

We asked if the same trends of performance would hold for the rat hippocampus dataset, which was characterized by very low mean firing rates but strong effect sizes. All methods were trained on a list of squared distances to a grid of place fields and on and the rat head orientation, as described in methods. Far more even than the neocortical data, neurons were described much better by XGBoost and the ensemble method than by the GLM (Fig. 7b). Many neurons shifted from being completely unpredictable by the GLM (pseudo-R^2 near zero) to very predictable by XGBoost and the ensemble (pseudo-R^2 above 0.2). These neurons thus have responses that do not correlate with firing in any one Gaussian place field. We note that the neural network performed poorly, likely due to the very low firing rates of most hippocampal cells (Supp. Fig. 2). The median spike rate of the 58 neurons in the dataset was just 0.2 spikes/s, and it was only on the four neurons with rates above 1 spikes/s that the neural network achieved pseudo-R^2 scores comparable to the GLM. The relative success of XGBoost was interesting given the failure of the neural network, and supported the general observation that boosted trees can work well with smaller and sparser datasets than those that neural networks generally require (Supp. Fig. 3). Thus for hippocampal cells, a method leveraging decision trees such as XGBoost or the ensemble is able to capture more structure in the neural response and thus demonstrate a deficiency of the parameterization of the GLM.

Discussion

We analyzed the ability of various machine learning techniques at the task of predicting binned spike counts in three brain regions. We found that of the tested ML methods, XGBoost and the ensemble routinely predicted spike counts more accurately than did the GLM, which is a popular method for neural data. Feedforward neural networks did not always outperform the GLM and were often worse than XGBoost and the ensemble. Machine learning methods, especially LSTMs, also outperformed GLMs when covariate and spike history were included as inputs. The ML methods performed comparably well with and without feature engineering, even for the very low spike rates of the hippocampus dataset. These findings indicate that a standard ML approach can serve as a reliable benchmark to test if data meets the assumptions of a GLM. Furthermore, it may be quite common that standard ML outperforms GLMs given standard feature choices.

When a GLM fails to explain data as well as more expressive, nonlinear methods, the current parameterization of inputs must relate to the data with a different nonlinearity than is assumed by the GLM. Such situations have been identified several times in the literature (Butts et al., 2011; Freeman et al., 2015; Heitman et al., 2016; McIntosh et al., 2016a). This unaccounted nonlinearity may produce feature weights that do not reflect true feature importance. A GLM will incorrectly predict no dependence on feature x whatsoever, for example, in the extreme case when the neural response to some feature x does not correlate with exp(x). The only way to ensure that feature weights can be reliably interpreted is to find an input parameterization that maximizes the GLM's predictive power. ML methods can assist this process by indicating how much nonlinearity remains to be explained. New features can then be tested, such as those suggested by a search for maximally informative dimensions (T. Sharpee, Rust, & Bialek, 2004). In our analysis, then, the GLM underperforms because we have selected the suboptimal input features. It is always theoretically possible to linearize features such that a GLM obtains equal predictive power. ML methods can highlight the deficiency of features that might have otherwise seemed uncontroversial. When applying a GLM or any simple model to neural data, it is important to compare its predictive power with standard ML methods to ensure the neural response is properly understood.

There are other ways of estimating the performance of a method besides benchmark nonlinear methods. For example, if the same exact stimulus can be given many times in a row, then we can estimate neural variability without having to model how activity depends on stimulus features (Schoppe, Harper, Willmore, King, & Schnupp, 2016). This approach, however, requires that we can model how neural responses vary with repetition (Grill-Spector, Henson, & Martin, 2006). This approach also makes it difficult to include spike history as an input, since the exact history is rarely repeated. We note that in some cases it may also be impossible to show the same stimulus multiple times, e.g. because eyes move. However, comparing these two classes of benchmark would be interesting on applications where both are feasible.

Advanced ML methods are not widely considered to be interpretable. Interpretation is not necessary for performance benchmarks, but it would be desirable to use these methods as standalone encoding models. We can better discuss this issue with a more precise definition of interpretability. Following Lipton, we make the distinction between a method's post-hoc interpretability, the ease of justifying its predictions, and transparency, the degree to which its operation and internal parameters are human-readable or easily understandable (Lipton et al., 2016). A GLM is certainly more transparent than many ML methods due to its algorithmic simplicity. Certain nonlinear extensions of the GLM have also been designed to remain transparent (Latimer et al., 2014; Maheswaranathan et al., 2017; McFarland et al., 2013; Theis et al., 2013; Williamson et al., 2015). For high-level areas, though, such as V4, the linearized features may be difficult to be interpreted themselves (Yamins et al., 2014), though it may be possible to increase the interpretability of features (Kaardal, Fitzgerald, Berry, & Sharpee, 2013). A GLM is also generally more conducive to post-hoc interpretations, though this is also possible with modern ML methods. It is possible, for example, to visualize the aspects of stimuli that most elicit a predicted response, as has been implemented in previous applications of neural networks to spike prediction (Lau, Stanley, & Dan, 2002; Prenger, Wu, David, & Gallant, 2004). Various other methods exist in the literature to enable post-hoc explanations (McAuley & Leskovec, 2013; Simonyan, Vedaldi, & Zisserman, 2013). Here we highlight Local Interpretable Model-Agnostic Explanations (LIME), an approach that fits simple models in the vicinity of single examples to allow a local interpretation (Ribeiro, Singh, & Guestrin, 2016). On problems where interpretability is important, such capabilities for post-hoc justifications may prove sufficient.

Not all types of interpretability are necessary for a given task, and many scientific questions can be answered based on predictive ability alone. Questions of the form, "does feature x contribute to neural activity?", for example, or "is past activity necessary to explain current activity?" require no method transparency. One can simply ask whether predictive power increases with feature x's inclusion or decreases upon its exclusion. Importance measures based on inclusion and exclusion, or upon the strategy of shuffling a covariate of interest, are well-studied in statistics and machine learning (Bell & Wang, 2000; Strobl, Boulesteix, Kneib, Augustin, & Zeileis, 2008). Depending on the application, it may thus be worthwhile to ask not just whether different features could improve a GLM but also whether it is enough to use ML methods directly. It is possible for many questions to stay agnostic to the form of linearized features and directly use changes in predictive ability.

With ongoing progress in machine learning, many standard techniques are easy to implement and can even be automated. Ensemble methods, for example, remove the need to choose any one algorithm. Moreover, the choice of model-specific parameters is made easy by hyperparameter search methods and optimizers. We hope that this ease of use might encourage use in the neurosciences, thereby increasing the power and efficiency of studies involving neural prediction without requiring complicated, application-specific methods development (e.g. (Corbett, Perreault, & Körding, 2012)). Community-supported projects in automated machine learning, such as autoSklearn and auto-Weka, are quickly improving and promise to handle the entire regression workflow (Feurer et al., 2015; Kotthoff, Thornton, Hoos, Hutter, & Leyton-Brown, 2016). Applied to neuroscience, these tools will allow researchers to gain descriptive power over current methods even with simple, out-of-the-box implementations.

Machine learning methods perform quite well and make minimal assumptions about the form of neural encoding. Models that seek to understand the form of the neural code can test if they systematically misconstrue the relationship between stimulus and response by comparing their performance to these benchmarks. Encoding models built with machine learning can thus greatly aid the construction of models that capture arbitrary nonlinearity and more accurately describe neural activity.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., . . . Devin, M. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467.*

Aljadeff, J., Lansdell, B. J., Fairhall, A. L., & Kleinfeld, D. (2016). Analysis of neuronal spike trains, deconstructed. *Neuron, 91*(2), 221-259.

Amirikian, B., & Georgopulos, A. P. (2000). Directional tuning profiles of motor cortical cells. *Neuroscience research, 36*(1), 73-79.

BayesianOptimization. (2016). *GitHub repository*. Retrieved from https://github.com/fmfn/BayesianOptimization

Bell, D. A., & Wang, H. (2000). A formalism for relevance and its application in feature subset selection. *Machine learning, 41*(2), 175-195.

Brown, E. N., Frank, L. M., Tang, D., Quirk, M. C., & Wilson, M. A. (1998). A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells. *The Journal of Neuroscience, 18*(18), 7411-7425.

Butts, D. A., Weng, C., Jin, J., Alonso, J.-M., & Paninski, L. (2011). Temporal precision in the visual pathway through the interplay of excitation and stimulus-driven suppression. *Journal of Neuroscience, 31*(31), 11313-11327.

Cameron, A. C., & Windmeijer, F. A. (1997). An R-squared measure of goodness of fit for some common nonlinear regression models. *Journal of Econometrics, 77*(2), 329-342.

Chen, T., & Guestrin, C. (2016a). Xgboost: A scalable tree boosting system. *arXiv preprint arXiv:1603.02754*.

Chichilnisky, E. (2001). A simple white noise analysis of neuronal light responses. *Network: Computation in Neural Systems, 12*(2), 199-213.

Chollet, F. (2015). keras. *GitHub repository*. Retrieved from https://github.com/fchollet/keras

Corbett, E. A., Perreault, E. J., & Körding, K. P. (2012). Decoding with limited neural data: a mixture of time-warped trajectory models for directional reaches. *Journal of neural engineering, 9*(3), 036002.

Domencich, T. A., & McFadden, D. (1975). *Urban travel demand-a behavioral analysis*. Retrieved from

Fernandes, H. L., Stevenson, I. H., Phillips, A. N., Segraves, M. A., & Kording, K. P. (2014). Saliency and saccade encoding in the frontal eye field during natural scene search. *Cerebral Cortex, 24*(12), 3232-3245.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). *Efficient and robust automated machine learning*. Paper presented at the Advances in Neural Information Processing Systems.

Freeman, J., Field, G. D., Li, P. H., Greschner, M., Gunning, D. E., Mathieson, K., . . . Simoncelli, E. P. (2015). Mapping nonlinear receptive field structure in primate retina at single cone resolution. *ELife, 4*.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics, 28*(2), 337-407.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189-1232.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural computation, 12*(10), 2451-2471.

Gerwinn, S., Macke, J. H., & Bethge, M. (2010). Bayesian inference for generalized linear models for spiking neurons. *Frontiers in Computational Neuroscience, 4*(12).

Grill-Spector, K., Henson, R., & Martin, A. (2006). Repetition and the brain: neural models of stimulus-specific effects. *Trends in cognitive sciences, 10*(1), 14-23.

Heitman, A., Brackbill, N., Greschner, M., Sher, A., Litke, A. M., & Chichilnisky, E. (2016). Testing pseudo-linear models of responses to natural scenes in primate retina. *bioRxiv*, 045336.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence, 20*(8), 832-844.

Kaardal, J., Fitzgerald, J. D., Berry, M. J., & Sharpee, T. O. (2013). Identifying functional bases for multidimensional neural computations. *Neural computation, 25*(7), 1870-1890.

Kaggle Winner's Blog. (2016). Retrieved from http://blog.kaggle.com/category/winners-interviews/

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., & Leyton-Brown, K. (2016). Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *Journal of Machine Learning Research, 17*, 1-5.

Latimer, K. W., Chichilnisky, E., Rieke, F., & Pillow, J. W. (2014). *Inferring synaptic conductances from spike trains with a biophysically inspired point process model*. Paper presented at the Advances in Neural Information Processing Systems.

Lau, B., Stanley, G. B., & Dan, Y. (2002). Computational subunits of visual cortical neurons revealed by artificial neural networks. *Proceedings of the National Academy of Sciences, 99*(13), 8974-8979.

Lipton, Z. C., Kale, D. C., Elkan, C., Wetzell, R., Vikram, S., McAuley, J., . . . Wang, C.-I. (2016). The Mythos of Model Interpretability. *IEEE Spectrum*.

Maheswaranathan, N., Baccus, S. A., & Ganguli, S. (2017). Inferring hidden structure in multilayered neural circuits. *bioRxiv*, 120956.

McAuley, J., & Leskovec, J. (2013). *Hidden factors and hidden topics: understanding rating dimensions with review text*. Paper presented at the Proceedings of the 7th ACM conference on Recommender systems.

McFarland, J. M., Cui, Y., & Butts, D. A. (2013). Inferring nonlinear neuronal computation based on physiologically plausible inputs. *PLoS Computational Biology, 9*(7), e1003143.

McIntosh, L., Maheswaranathan, N., Nayebi, A., Ganguli, S., & Baccus, S. (2016a). *Deep learning models of the retinal response to natural scenes*. Paper presented at the Advances in Neural Information Processing Systems.

Mizuseki, K., Sirota, A., Pastalkova, E., & Buzsáki, G. (2009a). Multi-unit recordings from the rat hippocampus made during open field foraging. Retrieved from http://dx.doi.org/10.6080/K0Z60KZ9

Mizuseki, K., Sirota, A., Pastalkova, E., & Buzsáki, G. (2009b). Theta oscillations provide temporal windows for local circuit computation in the entorhinal-hippocampal loop. *Neuron, 64*(2), 267-280.

Nelder, J. A., & Baker, R. J. (1972). Generalized linear models. *Encyclopedia of statistical sciences*.

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

Paninski, L., Fellows, M. R., Hatsopoulos, N. G., & Donoghue, J. P. (2004). Spatiotemporal tuning of motor cortical neurons for hand position and velocity. *Journal of Neurophysiology, 91*(1), 515-532.

Paninski, L., Shoham, S., Fellows, M. R., Hatsopoulos, N. G., & Donoghue, J. P. (2004). Superlinear population encoding of dynamic hand trajectory in primary motor cortex. *The Journal of Neuroscience, 24*(39), 8551-8561.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*(Oct), 2825-2830.

Pillow, J. W., Paninski, L., Uzzell, V. J., Simoncelli, E. P., & Chichilnisky, E. (2005). Prediction and decoding of retinal ganglion cell responses with a probabilistic spiking model. *The Journal of Neuroscience, 25*(47), 11003-11013.

Pillow, J. W., Shlens, J., Paninski, L., Sher, A., Litke, A. M., Chichilnisky, E., & Simoncelli, E. P. (2008). Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature, 454*(7207), 995-999.

Prenger, R., Wu, M. C.-K., David, S. V., & Gallant, J. L. (2004). Nonlinear V1 responses to natural scenes revealed by neural network analysis. *Neural Networks, 17*(5), 663-679.

Prud'homme, M., & Kalaska, J. F. (1994). Proprioceptive activity in primate primary somatosensory cortex during active arm reaching movements. *Journal of Neurophysiology, 72*(5), 2280-2301.

Ramkumar, P., Jas, M., Achakulvisut, T., Idrizović, A., themantalope, Acuna, D. E., . . . Dyer, E. (2017). Pyglmnet 1.0.1. doi:10.5281/zenodo.291992

Ramkumar, P., Lawlor, P. N., Glaser, J. I., Wood, D. K., Phillips, A. N., Segraves, M. A., & Kording, K. P. (2016). Feature-based attention and spatial selection in frontal eye fields during natural scene search. *Journal of Neurophysiology*, jn. 01044.02015.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). *Why should i trust you?: Explaining the predictions of any classifier*. Paper presented at the Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Saleh, M., Takahashi, K., & Hatsopoulos, N. G. (2012). Encoding of coordinated reach and grasp trajectories in primary motor cortex. *The Journal of Neuroscience, 32*(4), 1220-1232.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks, 61*, 85-117.

Schoppe, O., Harper, N. S., Willmore, B. D., King, A. J., & Schnupp, J. W. (2016). Measuring the performance of neural models. *Frontiers in Computational Neuroscience, 10*, 10.

Schwartz, O., Pillow, J. W., Rust, N. C., & Simoncelli, E. P. (2006). Spike-triggered neural characterization. *Journal of vision, 6*(4), 13-13.

Sharpee, T., Rust, N. C., & Bialek, W. (2004). Analyzing neural responses to natural signals: maximally informative dimensions. *Neural computation, 16*(2), 223-250.

Simoncelli, E. P., Paninski, L., Pillow, J., & Schwartz, O. (2004). Characterization of neural responses with stochastic stimuli. *The cognitive neurosciences, 3*, 327-338.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). *Practical bayesian optimization of machine learning algorithms*. Paper presented at the Advances in Neural Information Processing Systems.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15*(1), 1929-1958.

Stevenson, I. H., Cherian, A., London, B. M., Sachs, N. A., Lindberg, E., Reimer, J., . . . Kording, K. P. (2011). Statistical assessment of the stability of neural movement representations. *Journal of Neurophysiology, 106*(2), 764-774.

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., & Zeileis, A. (2008). Conditional variable importance for random forests. *BMC bioinformatics, 9*(1), 307.

Team, T. T. D., Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., . . . Belikov, A. (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*.

Theis, L., Chagas, A. M., Arnstein, D., Schwarz, C., & Bethge, M. (2013). Beyond GLMs: a generative mixture modeling approach to neural system identification. *PLoS Comput Biol, 9*(11), e1003356.

Truccolo, W., Eden, U. T., Fellows, M. R., Donoghue, J. P., & Brown, E. N. (2005). A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of Neurophysiology, 93*(2), 1074-1089.

Weber, A. I., & Pillow, J. W. (2016). Capturing the dynamical repertoire of single neurons with generalized linear models. *arXiv preprint arXiv:1602.07389*.

Williamson, R. S., Sahani, M., & Pillow, J. W. (2015). The equivalence of information-theoretic and likelihood-based methods for neural dimensionality reduction. *PLoS Computational Biology, 11*(4), e1004141.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks, 5*(2), 241-259.

Wu, M. C.-K., David, S. V., & Gallant, J. L. (2006). Complete functional characterization of sensory neurons by system identification. *Annu. Rev. Neurosci., 29*, 477-505.

Yamins, D. L., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., & DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences, 111*(23), 8619-8624.

Chapter 4: (Interlude) The Four Roles of Supervised Machine Learning in Systems Neuroscience

Foreword

Machine learning plays multiple roles in this dissertation. These roles are logically distinct, and represent the general roles that machine learning can play more broadly in neuroscience. Here, I describe these roles and review the literature in neuroscience in which machine learning plays each role. This chapter is reproduced from a review paper I co-authored, now published at *Progress in Neurobiology*[4]. The aim of this review was to describe the potential uses so that other researchers might take a similar approach.

Role 1 is to help create solutions to engineering problems. Chapter 1 features machine learning in this role, as do papers to which I contributed as a second author, especially Glaser et al. *(2020)*[5] (see Publications).

Role 2 is to help in identifying variables that are predictive of something, like neural activity or disease. This is role is played only by papers I contributed to as a middle author, especially Shen et al. (2020)[6] (see Publications).

Role 3 is to set benchmarks for simple models of the brain, as described in Chapter 2.

Role 4 is for machine learning to itself serve as a model for understanding the brain. This is exemplified by Chapters 3, 4, and 5.

[4] Glaser, Joshua I.*, Ari S. Benjamin*, Roozbeh Farhoodi*, and Konrad P. Kording. "The roles of supervised machine learning in systems neuroscience*". Progress in neurobiology*. 2019 Apr 1;175:126-37.
* denotes co-first authorship

[5] Glaser, Joshua I., **Ari S. Benjamin**, Raeed H. Chowdhury, Matthew G. Perich, Lee E. Miller, and Konrad P. Kording. "Machine learning for neural decoding." *Eneuro* 7, no. 4 (2020).

[6] Shen, Hanfei, Tony Liu, Jesse Cui, Piyush Borole, **Ari Benjamin**, Konrad Kording, and David Issadore. "A web-based automated machine learning platform to analyze liquid biopsy data." *Lab on a Chip* 20, no. 12 (2020)

Abstract

Over the last several years, the use of machine learning (ML) in neuroscience has been increasing exponentially. Here, we review ML's contributions, both realized and potential, across several areas of systems neuroscience. We describe four primary roles of ML within neuroscience: 1) creating solutions to engineering problems, 2) identifying predictive variables, 3) setting benchmarks for simple models of the brain, and 4) serving itself as a model for the brain. The breadth and ease of its applicability suggests that machine learning should be in the toolbox of most systems neuroscientists.

Introduction

There is a lot of enthusiasm about machine learning (ML). After all, it has allowed computers to surpass human-level performance at image classification (He et al. 2015), to beat humans in complex games such as "Go" (Silver et al. 2016), and to provide high-quality speech to text (Hannun et al. 2014) in popular mobile phones. Progress in ML is also getting attention in the scientific community. Writing in the July 2017 issue of Science focusing on "AI Transforms Science", editor Tim Appenzeller writes, "For scientists, prospects are mostly bright: AI promises to supercharge the process of discovery" (Appenzeller 2017).

The field of systems neuroscience is no exception. In the last few years there have been many opinion pieces about the importance of ML in neuroscience (Vu et al. 2018; Barak 2017; Paninski and Cunningham 2017; Vogt 2018; Hinton 2011). Moreover, when we analyze the number of journal articles about ML in neuroscience, we find that its use has been growing rapidly over the last 20 years (Fig. 1). Machine learning has been used in many different ways within this literature. In this review, we will catalog the breadth of conceptual applications of ML in systems neuroscience.

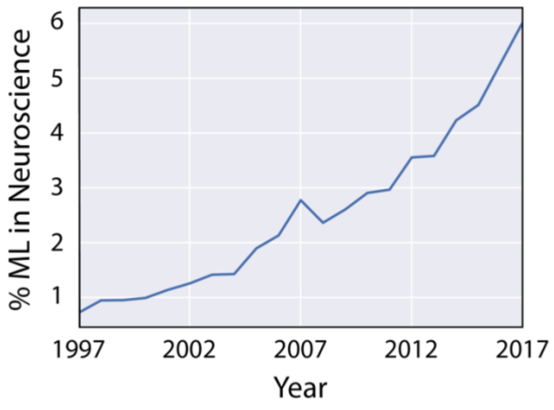


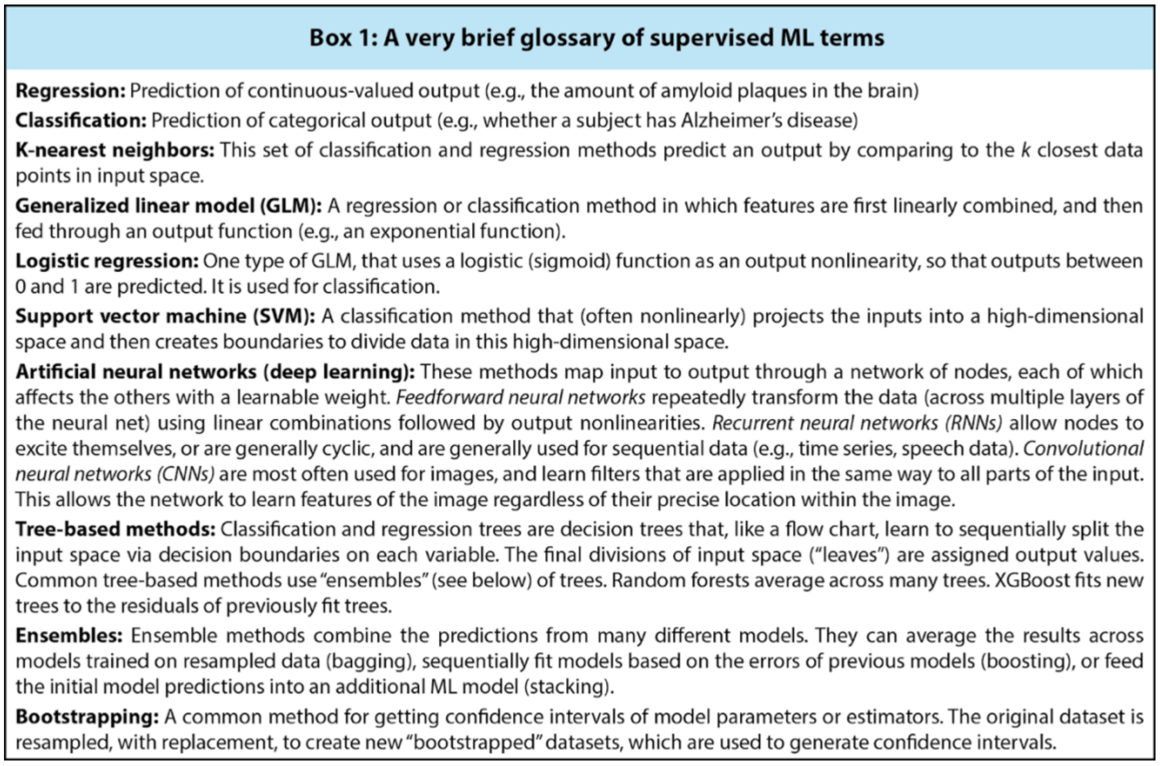
Figure 1: Growth of Machine Learning in Neuroscience. Here we plot the proportion of neuroscience papers that have used ML over the last two decades. The data represents the results of a search for “neuroscience” on Semantic Scholar. To identify the neuroscience papers that used machine learning methods we combined it with “machine learning”.

On the highest level, ML is typically divided into the subtypes of supervised, unsupervised, and reinforcement learning. Supervised learning builds a model that predicts outputs from input data. Unsupervised learning is concerned with finding structure in data, e.g. clustering, dimensionality reduction, and compression. Reinforcement learning allows a system to learn the best actions based on the reward that occurs at an end of a sequence of actions. This review focuses on supervised learning.

Why is creating progressively more accurate input/output mappings (i.e. regression or classification; see Box 1) worthy of a title like ‘The AI Revolution’ (Appenzeller 2017)? It is because countless questions can be framed in this manner. When classifying images, an input picture can be used to predict the object in the picture. When playing a game, the setup of the board (input) can be used to predict an optimal move (output). When texting on our smartphones, our current text is used to create suggestions of the next word. Science has many cases where data is measured and estimates need to be made.

In this review, we categorize the ways in which supervised ML promises to assist, or has already been applied to, problems in systems neuroscience. We believe that applications of supervised ML in this field can be divided in roughly four categories (Fig. 2). 1) *Solving engineering problems*. Machine learning can improve many methods for neuroscientists, such as medical diagnostics, brain-computer interfaces, and research tools. 2) *Identifying predictive variables*. Machine learning can more accurately determine whether variables (e.g., those related to the brain and outside world) predict each other. 3) *Benchmarking simple models*. We can compare the performance of simple interpretable models, to highly accurate ML models, to help determine the quality of the simple models. 4) *Serving as a model for the brain*. We can argue that the brain solves similar problems to ML systems, e.g. deep neural networks. The logic behind each of these applications is rather distinct.

For the bulk of this review, we will go into further detail about these four roles of ML in neuroscience. We will provide many examples, both realized and potential, of ML across several areas of systems neuroscience. More specifically, we will discuss the four roles of ML in relation to *neural function*, including neural activity and how it relates to behavior; and *neural structure*, i.e., neuroanatomy. Finally, we will discuss ML in practice (see Box 2), as that is crucial for useful applications.



Box 1: A very brief glossary of supervised ML terms

Regression: Prediction of continuous-valued output (e.g., the amount of amyloid plaques in the brain)

Classification: Prediction of categorical output (e.g., whether a subject has Alzheimer's disease)

K-nearest neighbors: This set of classification and regression methods predict an output by comparing to the *k* closest data points in input space.

Generalized linear model (GLM): A regression or classification method in which features are first linearly combined, and then fed through an output function (e.g., an exponential function).

Logistic regression: One type of GLM, that uses a logistic (sigmoid) function as an output nonlinearity, so that outputs between 0 and 1 are predicted. It is used for classification.

Support vector machine (SVM): A classification method that (often nonlinearly) projects the inputs into a high-dimensional space and then creates boundaries to divide data in this high-dimensional space.

Artificial neural networks (deep learning): These methods map input to output through a network of nodes, each of which affects the others with a learnable weight. *Feedforward neural networks* repeatedly transform the data (across multiple layers of the neural net) using linear combinations followed by output nonlinearities. *Recurrent neural networks (RNNs)* allow nodes to excite themselves, or are generally cyclic, and are generally used for sequential data (e.g., time series, speech data). *Convolutional neural networks (CNNs)* are most often used for images, and learn filters that are applied in the same way to all parts of the input. This allows the network to learn features of the image regardless of their precise location within the image.

Tree-based methods: Classification and regression trees are decision trees that, like a flow chart, learn to sequentially split the input space via decision boundaries on each variable. The final divisions of input space ("leaves") are assigned output values. Common tree-based methods use "ensembles" (see below) of trees. Random forests average across many trees. XGBoost fits new trees to the residuals of previously fit trees.

Ensembles: Ensemble methods combine the predictions from many different models. They can average the results across models trained on resampled data (bagging), sequentially fit models based on the errors of previous models (boosting), or feed the initial model predictions into an additional ML model (stacking).

Bootstrapping: A common method for getting confidence intervals of model parameters or estimators. The original dataset is resampled, with replacement, to create new "bootstrapped" datasets, which are used to generate confidence intervals.

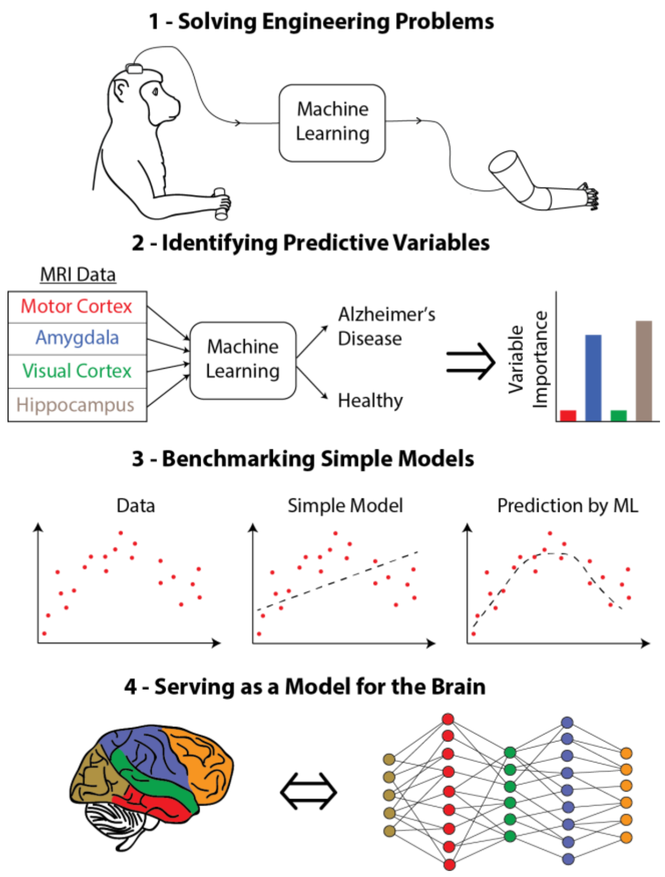


Figure 2: Examples of the four roles of supervised machine learning in neuroscience. 1 - *ML can solve engineering problems*. For example, it can help researchers control a prosthetic limb using brain activity. 2 - *ML can identify predictive variables*. For example, by using MRI data, we can identify which brain regions are most predictive for diagnosing Alzheimer's disease (Lebedev et al. 2014). 3 - *ML can benchmark simple models*. If the simple model's performance is close to that of the ML benchmark, it is more likely to be plausible. 4 - *ML can serve as a model of the brain*. For example, researchers have studied how neurons in the visual pathway correspond to units in an artificial network that is trained to classify images (Yamins and DiCarlo 2016).

Role 1: Solving engineering problems

A surprisingly wide array of engineering problems can be cast as prediction problems. Their common thread is that one would like to estimate some quantity of interest (Y) and can take measurements (X) that relate to that quantity. The relationship between X and Y, however, is unknown and might be complicated. We call these 'engineering problems' when the final quantity, Y, is all that is desired. In these problems, one does not need detailed understanding of the relationship - the aim is simply to estimate Y as accurately as possible. For example, email providers aim to determine whether an email is spam from the text of that email and only care about the prediction accuracy.

Traditionally, one would attempt to carefully understand this relationship, and synthesize this into a model. Modern machine learning (ML) is changing this paradigm. Instead of detailed expert knowledge of a process, a practitioner simply needs a large database of measurements along with the associated quantity of interest for each measurement. Machine learning algorithms can then automatically model their relationship. Once trained, an ML algorithm can make predictions for new measurements.

This 'engineering' framework is the traditional application of ML and is common in industry. In neuroscience, there are many problems that resemble this general problem format.

Neural Activity / Function

Many medical applications depend on successfully extracting information about intention, sensation, or disease from measurements of neural activity. One application is brain-computer interfaces (BCIs), which seek to use neural signals for control, such as for prosthetic limbs or computer cursors. While the form of neural activity, the 'neural code', may be unknown, it is possible to obtain large datasets of neural activity and the resulting movements, which enables ML predictions. Several groups have used modern ML techniques, such as recurrent neural networks, to improve BCI using spikes (Sussillo et al. 2016, 2012), ECoG (Elango et al. 2017), or EEG (Bashivan et al. 2015). In other applications, researchers used ML to predict disease state. For example, researchers predicted imminent seizures in epileptic patients using deep learning (Talathi 2017; Nigam and Graupe 2004) and ensemble methods (Brinkmann et al. 2016). Researchers have also classified neurological conditions, and there are several reviews on the subject (see (Arbabshirani et al. 2017) for classification using neuroimaging data, (Rathore et al. 2017) for classification of Alzheimer's disease, and (Vieira, Pinaya, and Mechelli 2017) for a focus on deep learning approaches). Predicting disease or intended movement from neural data is promising from a medical perspective.

An ML approach also promises to assist with the inverse of the above problem: predicting neural activity from variables in the outside world. Solving this problem is important if we want to use neural stimulation to induce accurate sensation. A prosthetic eye, for example, could be built by stimulating retinal ganglion cells in the correct manner according to the output of a camera (Nirenberg and Pandarinath 2012). The most accurate model of ganglion cell activity is currently a deep learning model trained to predict activity from naturalistic scenes (McIntosh et al. 2016). Similarly, prosthetic limbs could provide tactile and proprioceptive sensations if somatosensory neurons were correctly stimulated (Armenta Salas et al. 2018). Machine learning models may help to enable these neural prosthetics to induce sensations.

Machine learning can be used as a research tool to quantify behavior (D. J. Anderson and Perona 2014), such as movement, sleeping, and socializing. For example, we may want to quantify movement of the whole body using cheap video recordings. Recent progress in the field has made video quantification far more precise. Researchers have used deep learning to estimate human poses from video (Insafutdinov et al. 2016). Related approaches have recently gotten easier to use and less data intensive, and have been extended to animal tracking (Mathis et al. 2018; T. Pereira et al. 2018). Along with estimating poses, we can also directly estimate types of behavior (e.g., walking vs. stopping vs. jumping) from video (Kabra et al. 2013). Behavior can also be estimated based on other modalities such as audio recordings (Arthur et al. 2013). Progress in ML-based behavior tracking can help us understand neural control of behavior in more natural environments.

Finally, ML is helping the technical problem of obtaining accurate estimates of neural activity from raw measurements. Many imaging methods, such as EEG, MEG, and fMRI, require the solving of an 'inverse problem' - obtaining the source from the measurements. For example, researchers estimate the origin of EEG signals within the brain based on electrode recordings from the scalp. Recently, it has been observed that deep learning can improve the estimates for imaging (McCann, Jin, and Unser 2017). Neural networks have improved image denoising (Burger, Schuler, and Harmeling 2012; Xie, Xu, and Chen 2012) and deconvolution (L. Xu et al. 2014), can provide super-resolution images (Dong et al. 2016), and can even replace the entire image processing pipeline (Golkov et al. 2016; Vito et al. 2005). Outside of imaging, the deconvolution of time series data is a common example. For example, once a researcher has obtained traces of cellular calcium concentration, there is still the difficult 'inverse problem' of inferring the timing of the underlying spiking. Competition-style ML on labeled datasets provides good solutions (Berens et al. 2017). In each of these applications, a difficult engineering problem was replaced by building a large labeled dataset and learning the desired relationship.

Neuroanatomy / Structure

Along with diagnosing disease from neural activity, ML can also diagnose disease from neuroanatomy. For example, researchers can distinguish between Alzheimer's disease and healthy brains of older adults using MRI scans (Sarraf, Tofighi, and Others 2016). More generally, neuroanatomical measurements such as structural MRI and diffusion tensor imaging (DTI) can distinguish healthy from unhealthy patients across many conditions including schizophrenia, depression, autism and ADHD (Arbabshirani et al. 2017; Rathore et al. 2017; Vieira, Pinaya, and Mechelli 2017). Sometimes, ML enables surprising opportunities. For example, it has demonstrated using deep convolutional neural networks we can surprisingly predict cardiovascular risk factors from retinal fundus photographs (Poplin et al. 2018). The future will undoubtedly see ongoing efforts to automatically detect disease from biological data.

Machine learning also creates tools for neuroanatomists. The majority of research in neuroanatomy is based on imaging techniques. Thus, segmenting and labeling parts of an image, which usually requires manual annotation, is an especially important task. However, as imaging techniques improve and the volume of data increases, it will become infeasible to rely on manual annotation. To solve this problem, many ML techniques have been developed to automatically segment or label new images based on a dataset of previously labeled images. These techniques have been used to label medical images (Litjens et al. 2017; Ronneberger, Fischer, and Brox 2015). They have also been used to understand the connections and morphologies of neurons from electron microscopy (Helmstaedter et al. 2013; Funke et al. 2017; Turaga et al. 2010). As imaging data improve in resolution and volume, ML is becoming a crucial and even necessary tool for reconstructing and mapping neuroanatomy.

Role 2: Identifying predictive variables

Neuroscientists often ask questions of the form, “which variables are related to something of interest?” For example, which brain regions can predict each other? Which brain regions contain information related to a subject’s decision? Which cell types are affected by a certain disease? Machine learning (ML) can help to more accurately identify how informative one set of variables is about another. This is particularly instructive when there is a complex nonlinear relationship between the variables, which is often the case in neural systems. Answering these types of questions allows researchers to better understand the relationship between parts of the brain, stimuli, behavior, and more.

The general strategy resembles that of the engineering applications (Role 1). However, instead of only searching for maximal predictive accuracy, one examines which input variables lead to that accuracy. There are many methods to establish so-called ‘feature importance’ (also known as ‘feature selection’). Two of the simplest are the leave-one-out strategy, in which each variable is removed and one observes the decrease in accuracy, and the ‘best first’ strategy, in which the algorithm is run on each variable alone. The development of feature selection algorithms that appropriately allow for nonlinear feature interactions is an active field in ML and statistics (Tang, Alelyani, and Liu 2014). These methods allow us to get insights into which variables matter for a given problem.

A more traditional approach for this type of question would be to fit simple models to data, like linear regression, and to examine the coefficients of the fit. This approach is ubiquitous in science. Its basic drawback, however, is that one needs to assume a model, which may be inaccurate. For instance, linear regression will not provide meaningful coefficients and confidence intervals when the underlying relationship between inputs and outputs is nonlinear. The ML approach, on the other hand, seeks to maximize predictive accuracy and in doing so does not need to assume a simple functional form. This has the advantage that we can evaluate a variable’s importance even when the relationship between inputs and outputs is quite nonlinear. Plus, by bootstrapping, we can even find the confidence interval for their importance values. Machine learning combined with feature selection approaches can be universally applied to problems regardless of whether we know the underlying relationship.

Determining the important features can also help us to construct simpler models. Rather than using many inputs for a model, we can only use the important features as inputs. For example, determining which morphological features of neurons are most predictive of cell type can lead us to build more accurate generative models of morphologies (that are based on the most predictive features). Accurately determining the importance of features within ML algorithms is thus also beneficial for creating simpler models.

It is important to note that this approach examines the same variables that serve as raw inputs. Often the "features" we seek are different than the raw inputs. This is the case for vision, for example, in which the raw input variables may be as simple as single pixels. Finding these relevant features is a separate problem than the one we outline here in Role 2, and often involves looking within the "black box" of ML systems (described in Role 4) or using unsupervised learning methods (Guo et al. 2011; Suk et al. 2015; Längkvist, Karlsson, and Loutfi 2012).

Neural Activity / Function

Neuroscience has a long history of building encoding models, which aim to predict neural activity (e.g., spikes in an individual neuron, or BOLD signal in an fMRI voxel) based on variables in the outside world. This is a common approach to identifying the "role" of a brain area. The building of encoding models is a regression problem (from external variables to activity), and its purpose is more akin to feature importance than purely predictive power. This problem is an open invitation to use ML methods in combination with methods of determining feature importance.

Machine learning would not be necessary for encoding models if simpler methods were as accurate at describing neural activity. However, this is usually not the case. For example, we recently showed that XGBoost and ensemble methods led to significant performance improvements on datasets from motor cortex, somatosensory cortex, and hippocampus (Benjamin et al. 2017). These improvements were relative to Generalized Linear Models, which are ubiquitous in computational neuroscience. Others have also shown predictive improvements in other areas and modalities using methods such as

XGBoost (Viejo, Cortier, and Peyrache 2018) and deep learning (McIntosh et al. 2016; Klindt et al. 2017; Agrawal et al. 2014). These instances serve as warnings that although simple models may appear interpretable, they may be missing important aspects of how external variables relate to neural function.

Having improved encoding performance can more generally allow researchers to understand which covariates are predictive of neural activity. This generalizes the already common approach of adding additional variables to simple models and observing the increase in performance (e.g., (Stringer et al. 2018)). For example, the research on building encoding models of head-direction neurons using XGBoost (Viejo, Cortier, and Peyrache 2018) looked at the relative contribution of the different covariates (such as the direction of the head) within the encoding model. This allowed them to determine how the covariates mattered, without assuming the form of the relationship. Determining the importance of external variables is an important opportunity for modern ML methods.

The reverse problem, "what information can be read-out from activity from this brain area" can also answer questions about information content and the role of specific brain areas or cell types. For example, it is possible to decode from several brain areas during a perceptual discrimination task to determine their role (Hernández et al. 2010). As another example, we have compared decoding results from motor cortex from different task conditions to determine how uncertainty in the brain relates to different behavioral uncertainty (Dekleva et al. 2016; Glaser et al. 2018). The choice of decoding method has a large impact on performance. We have recently done a thorough test of different ML methods on datasets from motor cortex, somatosensory cortex, and hippocampus, and have shown that modern ML methods, such as neural networks and ensemble methods lead to increased decoding accuracy (Glaser et al. 2017). More accurate decoding can increase our understanding of how much information is contained in a neural population about another variable, such as a decision, movement, or location.

Researchers also want to understand the underlying factors that are predictive of a disease. This can be done by finding the importance of neuroimaging features in traditional classification techniques (e.g., determining which functional connectivity measures are predictive of Alzheimer's disease in a logistic regression classifier (Challis et al. 2015)). More recently, studies have used a variety of methods to look inside deep learning classifiers to determine the important features (e.g., determining fMRI connectivity relationships that are predictive of ADHD (Deshpande et al. 2015) and schizophrenia (Kim et al. 2016)). Determining the predictive factors of disease extends beyond neuroimaging data in humans. For instance, in a mouse model of depression, researchers determined which features of the connectivity between prefrontal cortex and limbic areas were predictive of pathological behavior (Hultman et al. 2016). They were then able to use this information to design a neural stimulation paradigm to restore normal behavior. Machine learning can prove to be a valuable tool to uncover predictive relationships between observables and disease across a wide range of neural activity modalities and diseases.

Neuroscience researchers often want to determine which variables matter for behavior, so that they can relate these variables to neural activity. We can apply ML to find what variables are predictive of behavior, without assuming the form of the relationship. For example, researchers have aimed to determine which visual features predict where we look next. This is a useful step in determining the neural basis of gaze control (Ramkumar et al. 2016). Traditionally, hand-designed visual features have been used to predict where we look next (Itti and Koch 2001), but recently researchers have more accurately predicted fixation locations using deep learning (Kümmerer, Theis, and Bethge 2014). As another example, researchers have studied how features in the environment predict the songs produced by male Drosophila during courtship (Coen et al. 2016). Using a generalized linear model, this study found that the distance between the male and female was the strongest predictor. This allowed the researchers to then investigate the neural pathway that was responsible for distance modulating song amplitude. More accurate behavioral models can allow researchers to better investigate the relationship between neural activity and behavior.

Neuroanatomy / Structure

Machine learning can help researchers better understand how neuroanatomical features across the brain are predictive of disease. A general approach is to construct an ML classifier to determine whether a subject has the disease, and then look at the importance of the features (e.g., brain areas or connections) in that classifier. In one example, researchers trained an SVM classifier to predict depression based on graph-theory based features derived from diffusion-weighted imaging, and then looked at the importance of those features (Sacchet et al. 2015). In another example, researchers trained a random forest classifier to predict Alzheimer's disease from structural MRI and then determined which brain areas were the most predictive features in this classifier (Lebedev et al. 2014). Another general approach is to compare classification models that are constructed using different features. For example, the previously mentioned paper (Lebedev et al. 2014) also compared classifiers constructed with different feature sets, e.g., one using cortical thickness measures and one using volumetric measures. There are thus multiple ways in which ML can inform us about the predictive relationship between neuroanatomic features and neurological disease.

Neurons have complicated shapes with varying biological structure and vary widely across brain regions and across species (Zeng and Sanes 2017). Many approaches have been proposed to classify neurons: electrophysiology (Markram et al. 2015; Teeter et al. 2018), morphology (Vasques et al. 2016), genetics or transcriptomics (Sümbül et al. 2014; Nelson, Sugino, and Hempel 2006; Tasic et al. 2016), and synaptic connectivity (Jonas and Kording 2015). Machine learning is often used for cell-type classification (Armañanzas and Ascoli 2015; Vasques et al. 2016). The cell types can be labeled based on one modality (e.g. whether the cell is inhibitory or excitatory), and then these labels can be predicted based on another modality (e.g., morphology). For instance, both (López-Cabrera and Lorenzo-Ginori 2017) and (Mihaljević et al. 2015) have used ML to predict cell type based on morphological features. This can both tell us which features are unique across cell types, and also which features are shared (Farhoodi and Kording 2018). In all of these areas, ML can help us to identify important features that shape neurons and transform our view of neuroanatomy.

Role 3: Benchmarking simple models

Machine learning (ML) cannot entirely replace simpler models within systems neuroscience. Such models embody human understanding and are necessary to test hypotheses. However, our understanding is meaningful only to the extent that these simple models are actually correct. While one can check a model's validity from its predictive performance (e.g. R^2), it is often hard to know how much error derives from sources of noise versus systematic insufficiencies of the model. This is where ML can help. It can serve as an approximate upper bound for how much structure a simpler model should explain. If a human-generated model is much less accurate than the ML benchmark, it is likely that important principles are missing or because the model is misguided. If, on the other hand, an intuitive model matches the performance of ML, it is more likely (but not guaranteed) that the posited concepts are, indeed, meaningful.

This approach stands in contrast to the current paradigm in which simple models are compared with previous (simple) models. This comparison may be meaningless if both models are very far from the peak ML predictive performance. Without a change in paradigm, we run the risk of not recognizing predictable complexity when it exists and not meaningfully advancing our understanding.

There are great examples of this type of benchmarking from outside of neuroscience. Intelligible models are necessary for healthcare, so that patient decisions can be based on these models. However, it is also important that the models are as accurate as possible. Thus, when researchers made new, interpretable, models of pneumonia risk and hospital readmission, they compared the performance of interpretable models with ML benchmarks (Caruana et al. 2015). In psychology, researchers have compared human-made models against ML benchmarks to understand the limitations of current behavioral models (Kleinberg, Liang, and Mullainathan 2017). This approach should also be advantageous within neuroscience.

Finally, we want to point out that models can be compared against benchmarks on subsets of the data, which can help researchers determine what aspects of their model need to be improved. As an abstract example, imagine that we have a simple model for the activity of a brain area during tasks A and B. The model is close to the ML benchmark for task A, but not B. This tells us that the model needs to be revised to better take task B into account. Thus, using benchmarks can also tell us which components of models need to be improved.

Neural Activity / Function

A common research narrative is to propose a new model (of neural activity or behavior) and show that it performs better than a previous simpler model. We believe that papers taking this ubiquitous approach should also compare a third model: an ML benchmark. This simple supplementary comparison could provide crucial information about how much neural activity or behavior remains to be explained. Unfortunately, benchmarking figures for new models are quite uncommon in neuroscience.

Recent work has demonstrated that ML benchmarks may somewhat embarrass simple models in neuroscience. Neural networks in particular, have been shown to often describe neural activity far better than traditional, simple models. Neural networks better predict the activity of retinal ganglion cells (McIntosh et al. 2016), primate V4 and IT (Yamins et al. 2014), and auditory cortex (Kell et al. 2018). While the case can be made that these networks are themselves models of brain function (as we do in Role 4), these results also are plain demonstrations of the deficiencies of previous models. It would be desirable if such checks were made with the introduction of new simple models.

Neuroanatomy / Structure

Machine learning can also help benchmark the simple models that describe neuroanatomy. For example, many models have been proposed to describe the complexity of neurons' morphologies. There are simple models describing the relationship between the diameters of segments at branching points (Rall 1964), the linear dependency of branch diameter on its length (Burke, Marks, and Ulfhake 1992), and the fractal dimensions of neurons and their self-similarity (Werner 2010). In all of these examples, it would be possible to use ML techniques on the raw data to create upper performance bounds for these models. This carries the promise to make anatomical modeling more meaningful.

Role 4: Serving as a model for the brain

The role of computational models of the brain is not only to predict, but also to serve as human-understandable distillations of how we think the brain works. It has recently become a more popular idea that deep neural networks are good models of the brain, albeit at a high level of abstraction (Marblestone, Wayne, and Kording 2016; Hassabis et al. 2017; Kietzmann, McClure, and Kriegeskorte 2017). Even a decade ago, this idea seemed less appealing to the field given the hyper-simplified form of contemporary neural networks. However, numerous recent empirical studies have pointed to unexpected parallels between the brain and neural networks trained on behaviorally relevant tasks, which we want to discuss here. While exciting, much work is needed (and is ongoing) to tighten the analogy between neural network models and the brain. Here we review these suggestive studies and discuss the various ways that artificial neural networks are becoming better models of biological ones. Neural activity and neuroanatomy are discussed without separation, as both aspects are often integrated together in ML models of the brain.

The trend of comparing trained neural networks to the brain was reignited recently due to the great achievements of neural networks at behavioral tasks, such as recognizing images (He et al. 2015). Interestingly, these networks have many parallels to the ventral stream in vision. These networks are explicitly hierarchical and multi-layered. Information from image pixels typically is processed through upwards of a dozen layers of "neurons", or nodes. In addition to their analogous organization, their activations are similar. For example, it has been observed that early nodes have Gabor-like receptive fields (Güçlü and van Gerven 2015), reminiscent of the edge detectors seen in V1. Moreover, activations in early/intermediate/later layers of these networks make excellent predictions of V1/V4/IT responses, respectively (of both individual neurons and fMRI responses) (Yamins and DiCarlo 2016; Yamins et al. 2014; Khaligh-Razavi and Kriegeskorte 2014; Güçlü and van Gerven 2015). Recent work has further extended the similarities. Deep neural networks are similarly invariant to viewpoint in object recognition (Saeed Reza Kheradpisheh et al. 2016), respond similarly across images (Khaligh-Razavi and Kriegeskorte 2014), and make similar types of errors (Saeed R. Kheradpisheh et al. 2016). This litany of similarities is longer and extends over a broader range of the visual cortex than any competing class of models.

The similarities between trained neural networks and the brain extend beyond the visual system. The format of these studies, nearly universally, is to compare the internal response properties of a brain area to those of a neural network trained on a behavioral task associated with that brain area. A pioneering study published three decades ago showed the similarity between posterior parietal neurons and a neural network trained to locate objects in a visual scene (Zipser and Andersen 1988). More recently, networks trained on scene recognition could accurately predict responses in the occipital place area (Bonner and Epstein 2018). Networks trained on speech recognition and musical genre prediction have activity similar to the auditory cortex (Kell et al. 2018). Recurrent neural networks trained to reproduce monkey movements contained units with activities very similar in selectivity to neurons in the primary motor cortex (Sussillo et al. 2015). The units of recurrent networks trained on navigation tasks have activations similar to the grid and place cells of the entorhinal cortex and hippocampus (Kanitscheider and Fiete 2017; Cueva and Wei 2018; Banino et al. 2018). The similarity between response properties of artificial neural networks and the brain is a sign that these models may capture important aspects of the brain's computations.

While these results could indicate that current networks are already good models of the brain, there are a few reasons to be skeptical (Lake et al. 2017). Several neural network architectures can all predict activity reasonably well, despite being different in form. Additionally, neural networks require large amounts of data to train, while the brain can often learn from few examples (Carey and Bartlett 1978; F. Xu and Tenenbaum 2007). Plus, artificial networks are plainly different in both architecture and response patterns from biological brains. An alternative explanation of these results is that typical patterns of tuning curves are emergent properties of any good distributed computing system operating on the real world. More work is needed to evaluate in which circumstances deep neural networks form good models of neural computation.

There is ongoing research to address the concern that artificial neural networks are not biologically plausible. One focus on creating biologically plausible neural networks is on having spiking (binary), as opposed to continuous, units. Many recent research papers have begun to create spiking neural networks that successfully solve typical machine learning (ML) problems (Zenke and Ganguli 2017; Nicola and Clopath 2017; Mozafari et al. 2018; Bellec et al. 2018). Another focus is on backpropagation, which is used for training neural nets, yet is not considered a biologically plausible mechanism for credit assignment. One recent paper showed that random feedback weights still allows for successful learning (Lillicrap et al. 2016), solving one of the implausible aspects of backpropagation. Other work has presented networks based on the apical/basal dendrites to solve the problem of credit assignment (Guerguiev, Lillicrap, and Richards 2017; Körding and König 2001; Sacramento et al. 2017). Plus, there have been many other recent (Scellier and Bengio 2016; Bengio et al. 2015, 2017) and historic (Hinton and McClelland 1988) works creating more plausible credit assignment mechanisms. However, one challenge is that many biologically-motivated deep learning algorithms do not scale well to large datasets (Bartunov et al. 2018). Finally, there has also been recent work developing architectures that are more biologically realistic (Costa et al. 2017; Linsley et al. 2018), for example those inspired by cortical microcircuits (Costa et al. 2017). Work on biologically-plausible deep learning will help to address how much artificial networks should be seen as faithful models of the brain.

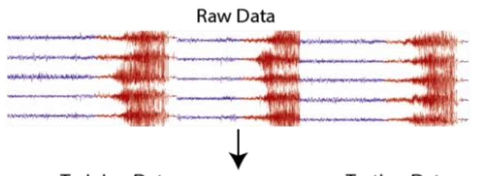
Another concern with these neural network models is interpretability. It would be worrisome to replace a brain we cannot meaningfully understand with a neural network equally as complicated. Still, it is advantageous that we can observe every hidden unit, weight, and activation within these models. There has been much recent work to develop methods to better understand what is occurring within neural networks. This includes methods for visualizing what features and information are represented at different scales within convolutional neural networks (Olah et al. 2018) and methods for understanding the dynamics within recurrent neural networks (Sussillo and Barak 2013). In fact, researchers are also developing new model architectures that are more easily interpretable (Foerster et al. 2017; Linderman et al. 2017). Neural network models can also be advantageous because researchers can do experiments on neural networks that would be impossible to perform on a biological brain. For example, researchers recently tested whether the tuning of individual units in neural networks were important for classification generalization (Morcos et al. 2018). Additionally, fully observable neural networks may serve as test-beds for new neural data analysis methods, which are greatly needed to increase understanding (Jonas and Kording 2017). While it is not guaranteed that lessons about neural networks will apply directly to the brain, findings can at least challenge assumptions within neuroscience and provide new hypotheses to test.

A final note about deep learning models of the brain is that the analogy invites a change in research focus. One might consider, for instance, focusing on which cost functions the brain is optimizing rather than the final properties of the network. Similarly, it is important to focus on determining the learning rules implemented by the brain. We have discussed some of these matters recently in a separate review, particularly focusing on how neuroscience should learn from machine learning's quest to understand neural networks (Marblestone, Wayne, and Kording 2016).

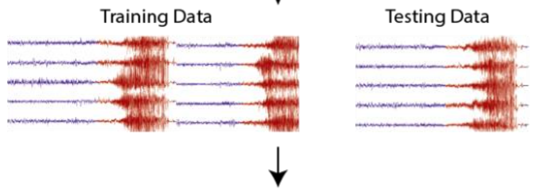
Although modeling the brain with neural networks is a popular approach, and perhaps most popular, other ML algorithms have also been proposed as models of the brain. Decision trees, for example, offer a compelling framework to model decision-making as a prunable tree of potential decisions (Huys et al. 2012). Simple neural circuits have also been modeled with threshold decision trees (Uchizawa, Douglas, and Maass 2006). The cerebellum and similar structures (like the mushroom body in *Drosophila*) can potentially be modeled as implementing a weighted nearest-neighbors calculation (Dasgupta, Stevens, and Navlakha 2017). Researchers have suggested that information processing in the brain relates to the ML concept of random projections (Arriaga and Vempala 2006). Plus, there have been many comparisons of the brain to reinforcement learning (Wang et al. 2018; Gläscher et al. 2010; Glimcher 2011; Doya 2000) and unsupervised learning (Hinton and Sejnowski 1999; Doya 2000), which we do not cover here. One may wonder if most ML models have been compared to the brain in some way.

Box 2: Example of Supervised ML in Practice

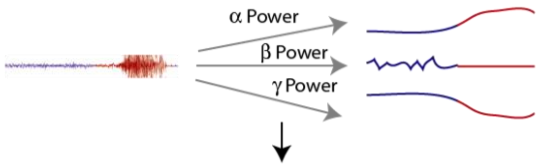
The data. To train a supervised ML method, one needs to have input data and labeled outputs. On the right, we show EEG traces, and the goal is to predict when a seizure is occurring (Brinkmann et al. 2016). The data has been labeled to show the output: blue for a normal brain state and red for a seizure.



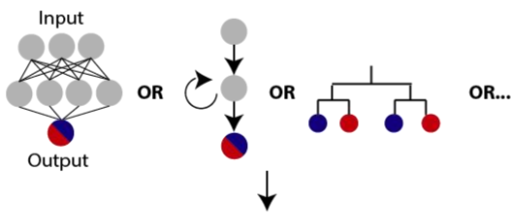
Split into training/testing tests. It is important to determine the accuracy of an ML model on separate data than was used to train the model. This is because a good model should generalize to new data and not just memorize training examples. A common technique is cross-validation, which repeatedly splits the data into different training/testing sets.



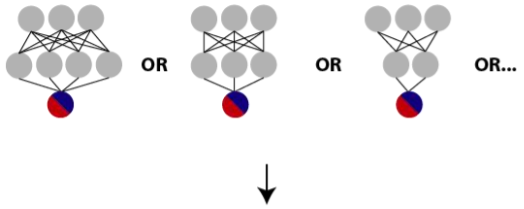
Preprocess / extract features from the data. Often, it is helpful to extract features from the raw data to use as inputs to the ML algorithm (e.g., for EEG, using power within multiple frequency bands rather than the full energy spectrum). Many modern ML techniques are making feature extraction less crucial. For example, raw images are often fed into convolutional neural nets. The same data preprocessing steps need to be performed on the training/testing sets.



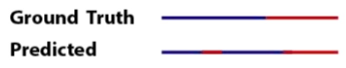
Model selection. There are many different ML algorithms, so which should we use? First, it depends on the data type. For instance, convolutional neural nets are good for images and recurrent neural nets are good for sequential data. It also depends on the amount of data. With limited data, models with fewer parameters are often helpful, although complex models can still be successful using techniques such as regularization. In general, we have found that tree-based methods (Random Forest and XGBoost) and feedforward neural nets with a couple hidden layers work well with default parameters for many problems. Finally, it is possible to not choose and use many algorithms (see ensembling below).



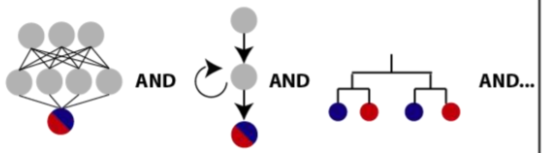
Hyperparameter optimization. Beyond the parameters that models fit, many ML models have hyperparameters, which generally relate to the model structure or how the model is fit. For example, neural networks can have different numbers of units in the hidden layer. The general strategy is to further split the training set into training and validation sets, and determine the hyperparameters that lead to the best fit on the validation set. Toolboxes exist that will intelligently find good hyperparameters (e.g., Hyperopt and BayesianOptimization).



Fit the model. Finally, we fit the ML model and make predictions on the test set. We can use our model for any of the 4 Roles, by determining accuracy, looking into the importance of input variables, and looking within the fitted model.



Ensembling. Ensemble methods are a common approach for maximizing performance that are often used in ML competitions, such as Kaggle. Ensembles combine predictions from multiple different ML models in order to leverage the benefits, and minimize the errors, from the different models.



AutoML. In recent years, there has been significant advances in automated ML. These packages automatically try to find the best models (and ensembles) for the problem, along with the best hyperparameters and data preprocessing steps.

Discussion

Here we have argued that supervised machine learning (ML) has four primary roles within system neuroscience: 1) Solving engineering problems; 2) Identifying predictive variables; 3) Benchmarking simple models; and 4) Serving as a model of the brain. As the current trend in applying ML to neuroscience (Fig. 1) indicates, we believe that the influence of ML on neuroscience will continue to grow.

Machine learning extends beyond supervised learning and thus can play an even greater role in neuroscience than we describe here. Using unsupervised learning methods, one can reduce the dimensionality of data (Cunningham and Yu 2014; Gao and Ganguli 2015), use clustering algorithms to discover new classes/categories (Armañanzas and Ascoli 2015; Drysdale et al. 2017), create generative models (Molano-Mazon et al. 2018; Arakaki, Barello, and Ahmadian 2017), and extract features automatically (Guo et al. 2011; Suk et al. 2015; Längkvist, Karlsson, and Loutfi 2012). Moreover, reinforcement learning (RL) is another ML category used within neuroscience. RL is inspired by behavioral psychology and can be used as a model of the brain to understand the mechanisms of reward-based learning and decision making (Wang et al. 2018; Gläscher et al. 2010; Glimcher 2011; Doya 2000). The four roles of this paper are all about supervised learning and do not include unsupervised learning and RL.

As neurotechnologies advance, the role of ML in neuroscience is likely to continue to grow. Datasets are rapidly growing (Stevenson and Kording 2011; Kandel et al. 2013) and becoming more complex (Glaser and Kording 2016; Vanwalleghem, Ahrens, and Scott 2018). Machine learning will be needed for this regime of data, as after all, there is only so much time a human being can spend looking at data. Moreover, as datasets get bigger, ML techniques become more accurate. Additionally, it is hard to reason about complex and high-dimensional datasets. Of all of the models that could explain a complex system, it is possible to think only about those models that are simple enough to imagine – to outline in human working memory. But in biology, as opposed to physics, there are good reasons to assume that truly meaningful models must be fairly complex (O'Leary, Sutton, and Marder 2015). While humans will correctly see some structure in the data, they will miss much of the actual structure. It is simply difficult to intuit models of nonlinear and recurrent biological systems. In these situations, it may be necessary to seek help from ML methods that can extract meaningful relationships from large datasets.

The use of ML will also continue to expand as ML gets easier to use. Applying ML has already become fairly straightforward. At application time, one requires a matrix of training features and a vector of known labels. Given the availability of the right software packages (Pedregosa et al. 2011), generally, only a few lines of code are then needed to train any ML system. In fact, there has been recent work on automated ML (Feurer et al. 2015; Kotthoff et al. 2017; Guyon et al. 2015), so that users do not need to make any decisions on specific methods to use, how to preprocess the data, or how to optimize hyperparameters. Thus, it is becoming less important for neuroscientists to know the details of the individual methods, which frees them to focus on the scientific questions that ML can answer.

The power of ML allows for the design of new types of experiments. Experiments will benefit from as much data as possible, measured both by the number of samples and number of dimensions. Since ML promotes experimental approaches that aim for predictions rather than for interpretations of the role of each variable, one can record as many variables as will improve predictions. This stands in contrast with the traditional scientific method of variable interpretation as, through multiple comparison testing corrections, it is not possible to say much about any variable if too many are recorded. Machine learning can also be used to optimize experimental design, e.g. by intelligently choosing stimuli that will maximize firing rates (Cowley et al. 2017) (here acting in Role 1 as an engineering tool). Researchers should keep ML methods in mind when designing experiments.

As is the case with any modeling, we wish to remind readers that it is important to be cautious when interpreting ML models. High predictability does not mean causality (Pearl 2009; Katz et al. 2016). This is especially true because there are so many unobserved variables within neuroscience. For instance, the apparent importance of a variable within a model may be inaccurate due to other unobserved variables (Stevenson 2018). Moreover, high model accuracy does not mean that it is actually a causally correct model of what is occurring in the brain. High accuracy is a necessary, but not a sufficient condition for model correctness. This is because there is an enormous space of potential models that could explain the data well. This is a difficulty of modeling the brain and identifying predictive variables with ML, but does not impact the use of ML for engineering, or benchmark, applications.

For a long time, neuroscientists have worked on improving ML techniques, and many ML techniques have been inspired by thoughts about brains and neural computation. With the growth of the ML field, the flow of information is becoming multi-directional. While neuroscience continues to inspire ML development, ML is also on the way to becoming one of the central tools and concepts in neuroscience.

References

Agrawal, Pulkit, Dustin Stansbury, Jitendra Malik, and Jack L. Gallant. 2014. "Pixels to Voxels: Modeling Visual Representation in the Human Brain." *arXiv [q-bio.NC]*. arXiv. http://arxiv.org/abs/1407.5104.

Anderson, David J., and Pietro Perona. 2014. "Toward a Science of Computational Ethology." *Neuron* 84 (1): 18–31.

Appenzeller, Tim. 2017. "The AI Revolution in Science." *Science*. https://doi.org/10.1126/science.aan7064.

Arakaki, Takafumi, G. Barello, and Yashar Ahmadian. 2017. "Capturing the Diversity of Biological Tuning Curves Using Generative Adversarial Networks." *arXiv [q-bio.QM]*. arXiv. http://arxiv.org/abs/1707.04582.

Arbabshirani, Mohammad R., Sergey Plis, Jing Sui, and Vince D. Calhoun. 2017. "Single Subject Prediction of Brain Disorders in Neuroimaging: Promises and Pitfalls." *NeuroImage* 145 (Pt B): 137–65.

Armañanzas, Rubén, and Giorgio A. Ascoli. 2015. "Towards the Automatic Classification of Neurons." *Trends in Neurosciences* 38 (5): 307–18.

Armenta Salas, Michelle, Luke Bashford, Spencer Kellis, Matiar Jafari, Hyeongchan Jo, Daniel Kramer, Kathleen Shanfield, et al. 2018. "Proprioceptive and Cutaneous Sensations in Humans Elicited by Intracortical Microstimulation." *eLife* 7 (April). https://doi.org/10.7554/eLife.32904.

Arriaga, Rosa I., and Santosh Vempala. 2006. "An Algorithmic Theory of Learning: Robust Concepts and Random Projection." *Machine Learning* 63 (2): 161–82.

Arthur, Benjamin J., Tomoko Sunayama-Morita, Philip Coen, Mala Murthy, and David L. Stern. 2013. "Multi-Channel Acoustic Recording and Automated Analysis of Drosophila Courtship Songs." *BMC Biology* 11 (January): 11.

Banino, Andrea, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, et al. 2018. "Vector-Based Navigation Using Grid-like Representations in Artificial Agents." *Nature*, May. https://doi.org/10.1038/s41586-018-0102-6.

Barak, Omri. 2017. "Recurrent Neural Networks as Versatile Tools of Neuroscience Research." *Current Opinion in Neurobiology* 46 (October): 1–6.

Bartunov, Sergey, Adam Santoro, Blake A. Richards, Geoffrey E. Hinton, and Timothy Lillicrap. 2018. "Assessing the Scalability of Biologically-Motivated Deep Learning Algorithms and Architectures." https://openreview.net/pdf?id=BypdvewVM.

Bashivan, Pouya, Irina Rish, Mohammed Yeasin, and Noel Codella. 2015. "Learning Representations from EEG with Deep Recurrent-Convolutional Neural Networks." *arXiv [cs.LG]*. arXiv. http://arxiv.org/abs/1511.06448.

Bellec, Guillaume, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. 2018. "Long Short-Term Memory and Learning-to-Learn in Networks of Spiking Neurons." *arXiv [cs.NE]*. arXiv. http://arxiv.org/abs/1803.09574.

Bengio, Yoshua, Dong-Hyun Lee, Jorg Bornschein, Thomas Mesnard, and Zhouhan Lin. 2015. “Towards Biologically Plausible Deep Learning.” *arXiv [cs.LG]*. arXiv. http://arxiv.org/abs/1502.04156.

Bengio, Yoshua, Thomas Mesnard, Asja Fischer, Saizheng Zhang, and Yuhuai Wu. 2017. “STDP-Compatible Approximation of Backpropagation in an Energy-Based Model.” *Neural Computation* 29 (3): 555–77.

Benjamin, Ari S., Hugo L. Fernandes, Tucker Tomlinson, Pavan Ramkumar, Chris VerSteeg, Lee Miller, and Konrad Paul Kording. 2017. “Modern Machine Learning Far Outperforms GLMs at Predicting Spikes.” *bioRxiv*. https://doi.org/10.1101/111450.

Berens, Philipp, Jeremy Freeman, Thomas Deneux, Nicolay Chenkov, Thomas McColgan, Artur Speiser, Jakob H. Macke, et al. 2017. “Community-Based Benchmarking Improves Spike Inference from Two-Photon Calcium Imaging Data.” *bioRxiv*. https://doi.org/10.1101/177956.

Bonner, Michael F., and Russell A. Epstein. 2018. “Computational Mechanisms Underlying Cortical Responses to the Affordance Properties of Visual Scenes.” *PLoS Computational Biology* 14 (4): e1006111.

Brinkmann, Benjamin H., Joost Wagenaar, Drew Abbot, Phillip Adkins, Simone C. Bosshard, Min Chen, Quang M. Tieng, et al. 2016. “Crowdsourcing Reproducible Seizure Forecasting in Human and Canine Epilepsy.” *Brain: A Journal of Neurology* 139 (Pt 6): 1713–22.

Burger, H. C., C. J. Schuler, and S. Harmeling. 2012. “Image Denoising: Can Plain Neural Networks Compete with BM3D?” In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/cvpr.2012.6247952.

Burke, R. E., W. B. Marks, and B. Ulfhake. 1992. “A Parsimonious Description of Motoneuron Dendritic Morphology Using Computer Simulation.” *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience* 12 (6): 2403–16.

Carey, Susan, and Elsa Bartlett. 1978. “Acquiring a Single New Word,” August. https://eric.ed.gov/?id=ED198703.

Caruana, Rich, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 2015. “Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-Day Readmission.” In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1721–30. KDD ’15. New York, NY, USA: ACM.

Challis, Edward, Peter Hurley, Laura Serra, Marco Bozzali, Seb Oliver, and Mara Cercignani. 2015. “Gaussian Process Classification of Alzheimer’s Disease and Mild Cognitive Impairment from Resting-State fMRI.” *NeuroImage* 112 (May): 232–43.

Coen, Philip, Marjorie Xie, Jan Clemens, and Mala Murthy. 2016. “Sensorimotor Transformations Underlying Variability in Song Intensity during Drosophila Courtship.” *Neuron* 89 (3): 629–44.

Costa, Rui, Ioannis Alexandros Assael, Brendan Shillingford, Nando de Freitas, and Tim Vogels. 2017. “Cortical Microcircuits as Gated-Recurrent Neural Networks.” In *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, 272–83. Curran Associates, Inc.

Cowley, Benjamin, Ryan Williamson, Katerina Clemens, Matthew Smith, and Byron M. Yu. 2017. “Adaptive Stimulus Selection for Optimizing Neural Population Responses.” In *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, 1396–1406. Curran Associates, Inc.

Cueva, Christopher J., and Xue-Xin Wei. 2018. “Emergence of Grid-like Representations by Training Recurrent Neural Networks to Perform Spatial Localization.” *arXiv [q-bio.NC]*. arXiv. http://arxiv.org/abs/1803.07770.

Cunningham, John P., and Byron M. Yu. 2014. “Dimensionality Reduction for Large-Scale Neural Recordings.” *Nature Neuroscience* 17 (11): 1500–1509.

Dasgupta, Sanjoy, Charles F. Stevens, and Saket Navlakha. 2017. “A Neural Algorithm for a Fundamental Computing Problem.” *Science* 358 (6364): 793–96.

Dekleva, Brian M., Pavan Ramkumar, Paul A. Wanda, Konrad P. Kording, and Lee E. Miller. 2016. “Uncertainty Leads to Persistent Effects on Reach Representations in Dorsal Premotor Cortex.” *eLife* 5 (July). https://doi.org/10.7554/eLife.14316.

Deshpande, Gopikrishna, Peng Wang, D. Rangaprakash, and Bogdan Wilamowski. 2015. “Fully Connected Cascade Artificial Neural Network Architecture for Attention Deficit Hyperactivity Disorder Classification From Functional Magnetic Resonance Imaging Data.” *IEEE Transactions on Cybernetics* 45 (12): 2668–79.

Dong, Chao, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2016. “Image Super-Resolution Using Deep Convolutional Networks.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38 (2): 295–307.

Doya, K. 2000. “Complementary Roles of Basal Ganglia and Cerebellum in Learning and Motor Control.” *Current Opinion in Neurobiology* 10 (6): 732–39.

Drysdale, Andrew T., Logan Grosenick, Jonathan Downar, Katharine Dunlop, Farrokh Mansouri, Yue Meng, Robert N. Fetcho, et al. 2017. “Resting-State Connectivity Biomarkers Define Neurophysiological Subtypes of Depression.” *Nature Medicine* 23 (1): 28–38.

Elango, Venkatesh, Aashish N. Patel, Kai J. Miller, and Vikash Gilja. 2017. “Sequence Transfer Learning for Neural Decoding.” *bioRxiv*. https://doi.org/10.1101/210732.

Farhoodi, R., and K. P. Kording. 2018. “Sampling Neuron Morphologies.” *bioRxiv*. biorxiv.org. https://www.biorxiv.org/content/early/2018/01/15/248385.abstract.

Feurer, Matthias, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. “Efficient and Robust Automated Machine Learning.” In *Advances in Neural Information Processing Systems 28*, edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, 2962–70. Curran Associates, Inc.

Foerster, Jakob N., Justin Gilmer, Jascha Sohl-Dickstein, Jan Chorowski, and David Sussillo. 2017. “Input Switched Affine Networks: An RNN Architecture Designed for Interpretability.” In *Proceedings of the 34th International Conference on Machine Learning*, edited by Doina Precup and Yee Whye Teh, 70:1136–45. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR.

Funke, Jan, Fabian David Tschopp, William Grisaitis, Arlo Sheridan, Chandan Singh, Stephan Saalfeld, and Srinivas C. Turaga. 2017. “A Deep Structured Learning Approach Towards Automating Connectome Reconstruction from 3D Electron Micrographs.” *arXiv [cs.CV]*. arXiv. http://arxiv.org/abs/1709.02974.

Gao, Peiran, and Surya Ganguli. 2015. “On Simplicity and Complexity in the Brave New World of Large-Scale Neuroscience.” *Current Opinion in Neurobiology* 32 (June): 148–55.

Gläscher, Jan, Nathaniel Daw, Peter Dayan, and John P. O’Doherty. 2010. “States versus Rewards: Dissociable Neural Prediction Error Signals Underlying Model-Based and Model-Free Reinforcement Learning.” *Neuron* 66 (4): 585–95.

Glaser, Joshua I., Raeed H. Chowdhury, Matthew G. Perich, Lee E. Miller, and Konrad P. Kording. 2017. “Machine Learning for Neural Decoding.” *arXiv [q-bio.NC]*. arXiv. http://arxiv.org/abs/1708.00909.

Glaser, Joshua I., and Konrad P. Kording. 2016. “The Development and Analysis of Integrated Neuroscience Data.” *Frontiers in Computational Neuroscience* 10 (February): 11.

Glaser, Joshua I., Matthew G. Perich, Pavan Ramkumar, Lee E. Miller, and Konrad P. Kording. 2018. “Population Coding of Conditional Probability Distributions in Dorsal Premotor Cortex.” *Nature Communications* 9 (1): 1788.

Glimcher, Paul W. 2011. “Understanding Dopamine and Reinforcement Learning: The Dopamine Reward Prediction Error Hypothesis.” *Proceedings of the National Academy of Sciences of the United States of America* 108 Suppl 3 (September): 15647–54.

Golkov, Vladimir, Alexey Dosovitskiy, Jonathan I. Sperl, Marion I. Menzel, Michael Czisch, Philipp Samann, Thomas Brox, and Daniel Cremers. 2016. “Q-Space Deep Learning: Twelve-Fold Shorter and Model-Free Diffusion MRI Scans.” *IEEE Transactions on Medical Imaging* 35 (5): 1344–51.

Güçlü, Umut, and Marcel A. J. van Gerven. 2015. “Deep Neural Networks Reveal a Gradient in the Complexity of Neural Representations across the Ventral Stream.” *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience* 35 (27): 10005–14.

Guerguiev, Jordan, Timothy P. Lillicrap, and Blake A. Richards. 2017. “Towards Deep Learning with Segregated Dendrites.” *eLife* 6 (December). https://doi.org/10.7554/eLife.22901.

Guo, Ling, Daniel Rivero, Julián Dorado, Cristian R. Munteanu, and Alejandro Pazos. 2011. “Automatic Feature Extraction Using Genetic Programming: An Application to Epileptic EEG Classification.” *Expert Systems with Applications* 38 (8): 10425–36.

Guyon, I., K. Bennett, G. Cawley, H. J. Escalante, S. Escalera, Tin Kam Ho, N. Macià, et al. 2015. “Design of the 2015 ChaLearn AutoML Challenge.” In *2015 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Hannun, Awni, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, et al. 2014. “Deep Speech: Scaling up End-to-End Speech Recognition.” *arXiv [cs.CL]*. arXiv. http://arxiv.org/abs/1412.5567.

Hassabis, Demis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. 2017. "Neuroscience-Inspired Artificial Intelligence." *Neuron* 95 (2): 245–58.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification." In *2015 IEEE International Conference on Computer Vision (ICCV)*. https://doi.org/10.1109/iccv.2015.123.

Helmstaedter, Moritz, Kevin L. Briggman, Srinivas C. Turaga, Viren Jain, H. Sebastian Seung, and Winfried Denk. 2013. "Connectomic Reconstruction of the Inner Plexiform Layer in the Mouse Retina." *Nature* 500 (7461): 168–74.

Hernández, Adrián, Verónica Nácher, Rogelio Luna, Antonio Zainos, Luis Lemus, Manuel Alvarez, Yuriria Vázquez, Liliana Camarillo, and Ranulfo Romo. 2010. "Decoding a Perceptual Decision Process across Cortex." *Neuron* 66 (2): 300–314.

Hinton, Geoffrey E. 2011. "Machine Learning for Neuroscience." *Neural Systems & Circuits* 1 (1). BioMed Central: 12.

Hinton, Geoffrey E., and James L. McClelland. 1988. "Learning Representations by Recirculation." In *Neural Information Processing Systems*, edited by D. Z. Anderson, 358–66. American Institute of Physics.

Hinton, Geoffrey E., and Terrence Joseph Sejnowski. 1999. *Unsupervised Learning: Foundations of Neural Computation*. MIT Press.

Hultman, Rainbo, Stephen D. Mague, Qiang Li, Brittany M. Katz, Nadine Michel, Lizhen Lin, Joyce Wang, et al. 2016. "Dysregulation of Prefrontal Cortex-Mediated Slow-Evolving Limbic Dynamics Drives Stress-Induced Emotional Pathology." *Neuron* 91 (2): 439–52.

Huys, Quentin J. M., Neir Eshel, Elizabeth O'Nions, Luke Sheridan, Peter Dayan, and Jonathan P. Roiser. 2012. "Bonsai Trees in Your Head: How the Pavlovian System Sculpts Goal-Directed Choices by Pruning Decision Trees." *PLoS Computational Biology* 8 (3): e1002410.

Insafutdinov, Eldar, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. 2016. "DeeperCut: A Deeper, Stronger, and Faster Multi-Person Pose Estimation Model." In *Computer Vision – ECCV 2016*, 34–50. Springer International Publishing.

Itti, L., and C. Koch. 2001. "Computational Modelling of Visual Attention." *Nature Reviews. Neuroscience* 2 (3): 194–203.

Jonas, Eric, and Konrad Kording. 2015. "Automatic Discovery of Cell Types and Microcircuitry from Neural Connectomics." *eLife* 4 (April): e04250.

Jonas, Eric, and Konrad Paul Kording. 2017. "Could a Neuroscientist Understand a Microprocessor?" *PLoS Computational Biology* 13 (1): e1005268.

Kabra, Mayank, Alice A. Robie, Marta Rivera-Alba, Steven Branson, and Kristin Branson. 2013. "JAABA: Interactive Machine Learning for Automatic Annotation of Animal Behavior." *Nature Methods* 10 (1): 64–67.

Kandel, Eric R., Henry Markram, Paul M. Matthews, Rafael Yuste, and Christof Koch. 2013. "Neuroscience Thinks Big (and Collaboratively)." *Nature Reviews. Neuroscience* 14 (9): 659–64.

Kanitscheider, Ingmar, and Ila Fiete. 2017. "Training Recurrent Networks to Generate Hypotheses about How the Brain Solves Hard Navigation Problems." In *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, 4529–38. Curran Associates, Inc.

Katz, Leor N., Jacob L. Yates, Jonathan W. Pillow, and Alexander C. Huk. 2016. "Dissociated Functional Significance of Decision-Related Activity in the Primate Dorsal Stream." *Nature* 535 (7611): 285–88.

Kell, Alexander J. E., Daniel L. K. Yamins, Erica N. Shook, Sam V. Norman-Haignere, and Josh H. McDermott. 2018. "A Task-Optimized Neural Network Replicates Human Auditory Behavior, Predicts Brain Responses, and Reveals a Cortical Processing Hierarchy." *Neuron* 98 (3): 630–44.e16.

Khaligh-Razavi, Seyed-Mahdi, and Nikolaus Kriegeskorte. 2014. "Deep Supervised, but Not Unsupervised, Models May Explain IT Cortical Representation." *PLoS Computational Biology* 10 (11): e1003915.

Kheradpisheh, Saeed Reza, Masoud Ghodrati, Mohammad Ganjtabesh, and Timothée Masquelier. 2016. "Deep Networks Can Resemble Human Feed-Forward Vision in Invariant Object Recognition." *Scientific Reports* 6 (September): 32672.

Kheradpisheh, Saeed R., Masoud Ghodrati, Mohammad Ganjtabesh, and Timothée Masquelier. 2016. "Humans and Deep Networks Largely Agree on Which Kinds of Variation Make Object Recognition Harder." *Frontiers in Computational Neuroscience* 10 (August): 92.

Kietzmann, Tim Christian, Patrick McClure, and Nikolaus Kriegeskorte. 2017. "Deep Neural Networks In Computational Neuroscience." *bioRxiv*. https://doi.org/10.1101/133504.

Kim, Junghoe, Vince D. Calhoun, Eunsoo Shim, and Jong-Hwan Lee. 2016. "Deep Neural Network with Weight Sparsity Control and Pre-Training Extracts Hierarchical Features and Enhances Classification Performance: Evidence from Whole-Brain Resting-State Functional Connectivity Patterns of Schizophrenia." *NeuroImage* 124 (Pt A): 127–46.

Kleinberg, Jon, Annie Liang, and Sendhil Mullainathan. 2017. "The Theory Is Predictive, but Is It Complete?: An Application to Human Perception of Randomness." In *Proceedings of the 2017 ACM Conference on Economics and Computation*, 125–26. EC '17. New York, NY, USA: ACM.

Klindt, David, Alexander S. Ecker, Thomas Euler, and Matthias Bethge. 2017. "Neural System Identification for Large Populations Separating 'what' and 'where.'" In *Advances in Neural Information Processing Systems*, 3509–19.

Körding, K. P., and P. König. 2001. "Supervised and Unsupervised Learning with Two Sites of Synaptic Integration." *Journal of Computational Neuroscience* 11 (3): 207–15.

Kotthoff, Lars, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. 2017. "Auto-WEKA 2.0: Automatic Model Selection and Hyperparameter Optimization in WEKA." *Journal of Machine Learning Research: JMLR* 18 (1). JMLR. org: 826–30.

Kümmerer, Matthias, Lucas Theis, and Matthias Bethge. 2014. "Deep Gaze I: Boosting Saliency Prediction with Feature Maps Trained on ImageNet." *arXiv [cs.CV]*. arXiv. http://arxiv.org/abs/1411.1045.

Lake, Brenden M., Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. 2017. "Building Machines That Learn and Think like People." *The Behavioral and Brain Sciences* 40 (January): e253.

Längkvist, Martin, Lars Karlsson, and Amy Loutfi. 2012. "Sleep Stage Classification Using Unsupervised Feature Learning." *Advances in Artificial Neural Systems* 2012: 1–9.

Lebedev, A. V., E. Westman, G. J. P. Van Westen, M. G. Kramberger, A. Lundervold, D. Aarsland, H. Soininen, et al. 2014. "Random Forest Ensembles for Detection and Prediction of Alzheimer's Disease with a Good between-Cohort Robustness." *NeuroImage. Clinical* 6 (August): 115–25.

Lillicrap, Timothy P., Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. 2016. "Random Synaptic Feedback Weights Support Error Backpropagation for Deep Learning." *Nature Communications* 7 (November): 13276.

Linderman, Scott, Matthew Johnson, Andrew Miller, Ryan Adams, David Blei, and Liam Paninski. 2017. "Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems." In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, edited by Aarti Singh and Jerry Zhu, 54:914–22. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR.

Linsley, Drew, Junkyung Kim, Vijay Veerabadran, and Thomas Serre. 2018. "Learning Long-Range Spatial Dependencies with Horizontal Gated-Recurrent Units." *arXiv [cs.CV]*. arXiv. http://arxiv.org/abs/1805.08315.

Litjens, Geert, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A. W. M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. 2017. "A Survey on Deep Learning in Medical Image Analysis." *Medical Image Analysis* 42 (December): 60–88.

López-Cabrera, José Daniel, and Juan Valentin Lorenzo-Ginori. 2017. "Automatic Classification of Traced Neurons Using Morphological Features." *Computación Y Sistemas* 21 (3). https://doi.org/10.13053/cys-21-3-2495.

Marblestone, Adam H., Greg Wayne, and Konrad P. Kording. 2016. "Toward an Integration of Deep Learning and Neuroscience." *Frontiers in Computational Neuroscience* 10 (September): 94.

Markram, Henry, Eilif Muller, Srikanth Ramaswamy, Michael W. Reimann, Marwan Abdellah, Carlos Aguado Sanchez, Anastasia Ailamaki, et al. 2015. "Reconstruction and Simulation of Neocortical Microcircuitry." *Cell* 163 (2): 456–92.

Mathis, Alexander, Pranav Mamidanna, Taiga Abe, Kevin M. Cury, Venkatesh N. Murthy, Mackenzie W. Mathis, and Matthias Bethge. 2018. "Markerless Tracking of User-Defined Features with Deep Learning." *arXiv [cs.CV]*. arXiv. http://arxiv.org/abs/1804.03142.

McCann, Michael T., Kyong Hwan Jin, and Michael Unser. 2017. "Convolutional Neural Networks for Inverse Problems in Imaging: A Review." *IEEE Signal Processing Magazine* 34 (6): 85–95.

McIntosh, Lane T., Niru Maheswaranathan, Aran Nayebi, Surya Ganguli, and Stephen A. Baccus. 2016. "Deep Learning Models of the Retinal Response to Natural Scenes." *Advances in Neural Information Processing Systems* 29: 1369–77.

Mihaljević, Bojan, Ruth Benavides-Piccione, Concha Bielza, Javier DeFelipe, and Pedro Larrañaga. 2015. “Bayesian Network Classifiers for Categorizing Cortical GABAergic Interneurons.” *Neuroinformatics* 13 (2): 193–208.

Molano-Mazon, Manuel, Arno Onken, Eugenio Piasini, and Stefano Panzeri. 2018. “Synthesizing Realistic Neural Population Activity Patterns Using Generative Adversarial Networks.” *arXiv [q-bio.NC]*. arXiv. http://arxiv.org/abs/1803.00338.

Morcos, Ari S., David G. T. Barrett, Neil C. Rabinowitz, and Matthew Botvinick. 2018. “On the Importance of Single Directions for Generalization.” *arXiv [stat.ML]*. arXiv. http://arxiv.org/abs/1803.06959.

Mozafari, Milad, Mohammad Ganjtabesh, Abbas Nowzari-Dalini, Simon J. Thorpe, and Timothée Masquelier. 2018. “Combining STDP and Reward-Modulated STDP in Deep Convolutional Spiking Neural Networks for Digit Recognition.” *arXiv [cs.CV]*. arXiv. http://arxiv.org/abs/1804.00227.

Nelson, Sacha B., Ken Sugino, and Chris M. Hempel. 2006. “The Problem of Neuronal Cell Types: A Physiological Genomics Approach.” *Trends in Neurosciences* 29 (6): 339–45.

Nicola, Wilten, and Claudia Clopath. 2017. “Supervised Learning in Spiking Neural Networks with FORCE Training.” *Nature Communications* 8 (1): 2208.

Nigam, Vivek Prakash, and Daniel Graupe. 2004. “A Neural-Network-Based Detection of Epilepsy.” *Neurological Research* 26 (1): 55–60.

Nirenberg, Sheila, and Chethan Pandarinath. 2012. “Retinal Prosthetic Strategy with the Capacity to Restore Normal Vision.” *Proceedings of the National Academy of Sciences of the United States of America* 109 (37): 15012–17.

Olah, Chris, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. 2018. “The Building Blocks of Interpretability.” *Distill* 3 (3). distill.pub: e10.

O’Leary, Timothy, Alexander C. Sutton, and Eve Marder. 2015. “Computational Models in the Age of Large Datasets.” *Current Opinion in Neurobiology* 32 (June): 87–94.

Paninski, Liam, and John Cunningham. 2017. “Neural Data Science: Accelerating the Experiment-Analysis-Theory Cycle in Large-Scale Neuroscience.” *bioRxiv*. https://doi.org/10.1101/196949.

Pearl, Judea. 2009. “Causal Inference in Statistics: An Overview.” *Statistics Surveys* 3 (0): 96–146.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. “Scikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research: JMLR* 12 (Oct): 2825–30.

Pereira, Talmo, Diego Aldarondo, Lindsay Willmore, Mikhail Kislin, Samuel S. Wang, Mala Murthy, and Joshua W. Shaevitz. 2018. “Fast Animal Pose Estimation Using Deep Neural Networks.” *bioRxiv*. https://doi.org/10.1101/331181.

Poplin, Ryan, Avinash V. Varadarajan, Katy Blumer, Yun Liu, Michael V. McConnell, Greg S. Corrado, Lily Peng, and Dale R. Webster. 2018. “Prediction of Cardiovascular Risk Factors from Retinal Fundus Photographs via Deep Learning.” *Nature Biomedical Engineering* 2 (3): 158–64.

Rall, W. 1964. “Theoretical Significance of Dendritic Trees for Neuronal Input-Output Relations.” *Neural Theory and Modeling*. Stanford University Press, 73–97.

Ramkumar, Pavan, Patrick N. Lawlor, Joshua I. Glaser, Daniel K. Wood, Adam N. Phillips, Mark A. Segraves, and Konrad P. Kording. 2016. “Feature-Based Attention and Spatial Selection in Frontal Eye Fields during Natural Scene Search.” *Journal of Neurophysiology* 116 (3): 1328–43.

Rathore, Saima, Mohamad Habes, Muhammad Aksam Iftikhar, Amanda Shacklett, and Christos Davatzikos. 2017. “A Review on Neuroimaging-Based Classification Studies and Associated Feature Extraction Methods for Alzheimer’s Disease and Its Prodromal Stages.” *NeuroImage* 155 (July): 530–48.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. 2015. “U-Net: Convolutional Networks for Biomedical Image Segmentation.” In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 234–41. Springer International Publishing.

Sacchet, Matthew D., Gautam Prasad, Lara C. Foland-Ross, Paul M. Thompson, and Ian H. Gotlib. 2015. “Support Vector Machine Classification of Major Depressive Disorder Using Diffusion-Weighted Neuroimaging and Graph Theory.” *Frontiers in Psychiatry / Frontiers Research Foundation* 6 (February): 21.

Sacramento, João, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. 2017. “Dendritic Error Backpropagation in Deep Cortical Microcircuits.” *arXiv [q-bio.NC]*. arXiv. http://arxiv.org/abs/1801.00062.

Sarraf, Saman, Ghassem Tofighi, and Others. 2016. “DeepAD: Alzheimer′ S Disease Classification via Deep Convolutional Neural Networks Using MRI and fMRI.” *bioRxiv*. Cold Spring Harbor Laboratory, 070441.

Scellier, Benjamin, and Yoshua Bengio. 2016. "Equilibrium Propagation: Bridging the Gap Between Energy-Based Models and Backpropagation." *arXiv [cs.LG]*. arXiv. http://arxiv.org/abs/1602.05179.

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, et al. 2016. "Mastering the Game of Go with Deep Neural Networks and Tree Search." *Nature* 529 (7587): 484–89.

Stevenson, Ian H. 2018. "Omitted Variable Bias in GLMs of Neural Spiking Activity." *bioRxiv*. https://doi.org/10.1101/317511.

Stevenson, Ian H., and Konrad P. Kording. 2011. "How Advances in Neural Recording Affect Data Analysis." *Nature Neuroscience* 14 (2): 139–42.

Stringer, C., M. Pachitariu, N. Steinmetz, and C. B. Reddy. 2018. "Spontaneous Behaviors Drive Multidimensional, Brain-Wide Population Activity." *bioRxiv*. biorxiv.org. https://www.biorxiv.org/content/early/2018/04/22/306019.abstract.

Suk, Heung-Il, Seong-Whan Lee, Dinggang Shen, and Alzheimer's Disease Neuroimaging Initiative. 2015. "Latent Feature Representation with Stacked Auto-Encoder for AD/MCI Diagnosis." *Brain Structure & Function* 220 (2): 841–59.

Sümbül, Uygar, Sen Song, Kyle McCulloch, Michael Becker, Bin Lin, Joshua R. Sanes, Richard H. Masland, and H. Sebastian Seung. 2014. "A Genetic and Computational Approach to Structurally Classify Neuronal Types." *Nature Communications* 5 (March): 3512.

Sussillo, David, and Omri Barak. 2013. "Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks." *Neural Computation* 25 (3): 626–49.

Sussillo, David, Mark M. Churchland, Matthew T. Kaufman, and Krishna V. Shenoy. 2015. "A Neural Network That Finds a Naturalistic Solution for the Production of Muscle Activity." *Nature Neuroscience* 18 (7): 1025–33.

Sussillo, David, Paul Nuyujukian, Joline M. Fan, Jonathan C. Kao, Sergey D. Stavisky, Stephen Ryu, and Krishna Shenoy. 2012. "A Recurrent Neural Network for Closed-Loop Intracortical Brain–machine Interface Decoders." *Journal of Neural Engineering* 9 (2): 026027.

Sussillo, David, Sergey D. Stavisky, Jonathan C. Kao, Stephen I. Ryu, and Krishna V. Shenoy. 2016. "Making Brain–machine Interfaces Robust to Future Neural Variability." *Nature Communications* 7: 13749.

Talathi, Sachin S. 2017. "Deep Recurrent Neural Networks for Seizure Detection and Early Seizure Detection Systems." *arXiv [q-bio.QM]*. arXiv. http://arxiv.org/abs/1706.03283.

Tang, Jiliang, Salem Alelyani, and Huan Liu. 2014. "Feature Selection for Classification: A Review." In *Data Classification: Algorithms and Applications*, edited by Charu Aggarwal, 37. CRC Press.

Tasic, Bosiljka, Vilas Menon, Thuc Nghi Nguyen, Tae Kyung Kim, Tim Jarsky, Zizhen Yao, Boaz Levi, et al. 2016. "Adult Mouse Cortical Cell Taxonomy Revealed by Single Cell Transcriptomics." *Nature Neuroscience* 19 (2): 335–46.

Teeter, Corinne, Ramakrishnan Iyer, Vilas Menon, Nathan Gouwens, David Feng, Jim Berg, Aaron Szafer, et al. 2018. "Generalized Leaky Integrate-and-Fire Models Classify Multiple Neuron Types." *Nature Communications* 9 (1): 709.

Turaga, Srinivas C., Joseph F. Murray, Viren Jain, Fabian Roth, Moritz Helmstaedter, Kevin Briggman, Winfried Denk, and H. Sebastian Seung. 2010. "Convolutional Networks Can Learn to Generate Affinity Graphs for Image Segmentation." *Neural Computation* 22 (2): 511–38.

Uchizawa, Kei, Rodney Douglas, and Wolfgang Maass. 2006. "On the Computational Power of Threshold Circuits with Sparse Activity." *Neural Computation* 18 (12): 2994–3008.

Vanwalleghem, Gilles C., Misha B. Ahrens, and Ethan K. Scott. 2018. "Integrative Whole-Brain Neuroscience in Larval Zebrafish." *Current Opinion in Neurobiology* 50 (February): 136–45.

Vasques, Xavier, Laurent Vanel, Guillaume Villette, and Laura Cif. 2016. "Morphological Neuron Classification Using Machine Learning." *Frontiers in Neuroanatomy* 10 (November): 102.

Vieira, Sandra, Walter H. L. Pinaya, and Andrea Mechelli. 2017. "Using Deep Learning to Investigate the Neuroimaging Correlates of Psychiatric and Neurological Disorders: Methods and Applications." *Neuroscience and Biobehavioral Reviews* 74 (Pt A): 58–75.

Viejo, Guillaume, Thomas Cortier, and Adrien Peyrache. 2018. "Brain-State Invariant Thalamo-Cortical Coordination Revealed by Non-Linear Encoders." *PLoS Computational Biology* 14 (3): e1006041.

Vito, Ernesto De, Lorenzo Rosasco, Andrea Caponnetto, Umberto De Giovannini, and Francesca Odone. 2005. "Learning from Examples as an Inverse Problem." *Journal of Machine Learning Research: JMLR* 6 (May): 883–904.

Vogt, Nina. 2018. "Machine Learning in Neuroscience." *Nature Methods* 15 (1). Nature Publishing Group: 33.

Vu, Mai-Anh T., Tülay Adalı, Demba Ba, György Buzsáki, David Carlson, Katherine Heller, Conor Liston, et al. 2018. “A Shared Vision for Machine Learning in Neuroscience.” *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience* 38 (7): 1601–7.

Wang, J. X., Z. Kurth-Nelson, D. Kumaran, and D. Tirumala. 2018. “Prefrontal Cortex as a Meta-Reinforcement Learning System.” *bioRxiv*. biorxiv.org. https://www.biorxiv.org/content/early/2018/04/06/295964.abstract.

Werner, Gerhard. 2010. “Fractals in the Nervous System: Conceptual Implications for Theoretical Neuroscience.” *Frontiers in Physiology* 1 (July): 15.

Xie, Junyuan, Linli Xu, and Enhong Chen. 2012. “Image Denoising and Inpainting with Deep Neural Networks.” In *Advances in Neural Information Processing Systems 25*, edited by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, 341–49. Curran Associates, Inc.

Xu, Fei, and Joshua B. Tenenbaum. 2007. “Word Learning as Bayesian Inference.” *Psychological Review* 114 (2): 245–72.

Xu, Li, Jimmy S. J. Ren, Ce Liu, and Jiaya Jia. 2014. “Deep Convolutional Neural Network for Image Deconvolution.” In *Advances in Neural Information Processing Systems 27*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, 1790–98. Curran Associates, Inc.

Yamins, Daniel L. K., and James J. DiCarlo. 2016. “Using Goal-Driven Deep Learning Models to Understand Sensory Cortex.” *Nature Neuroscience* 19 (3): 356–65.

Yamins, Daniel L. K., Ha Hong, Charles F. Cadieu, Ethan A. Solomon, Darren Seibert, and James J. DiCarlo. 2014. “Performance-Optimized Hierarchical Models Predict Neural Responses in Higher Visual Cortex.” *Proceedings of the National Academy of Sciences of the United States of America* 111 (23): 8619–24.

Zeng, Hongkui, and Joshua R. Sanes. 2017. “Neuronal Cell-Type Classification: Challenges, Opportunities and the Path Forward.” *Nature Reviews. Neuroscience* 18 (9): 530–46.

Zenke, Friedemann, and Surya Ganguli. 2017. “SuperSpike: Supervised Learning in Multi-Layer Spiking Neural Networks.” *arXiv [q-bio.NC]*. arXiv. http://arxiv.org/abs/1705.11146.

Zipser, D., and R. A. Andersen. 1988. “A Back-Propagation Programmed Network That Simulates Response Properties of a Subset of Posterior Parietal Neurons.” *Nature* 331 (6158): 679–84.

Chapter 5: Learning to infer in recurrent biological networks

Foreword

This chapter asks how the sensory cortex might learn representations of the external world. It is situated most closely to the field of artificial intelligence called, appropriately, *representation learning*. The goal of this chapter is to reconcile representation learning algorithms with biology, and thus act as hypotheses for sensory neuroscience. Broadly, it is an example of **Role 4**: machine learning as a model of the brain.

Representation learning is an old idea within AI. Here, a representation is defined as some transformation of sensory data into a new space. Principal Components Analysis (PCA), sparse coding, and k-means clustering are all examples of unsupervised representation learning algorithms. What is common to these algorithms is the existence of transformations both from data to a representation and from representation to data.

When taken as a model of the sensory cortex, one must adopt a central dogma that the brain, too, forms various representations and can transform information between them. These transformations are presumed to be more complicated than e.g. in PCA.

Temporarily bracketing the question of what these transformations might be, this chapter asks what neural circuits might be used to adjust and improve them towards their objective. This question is possible because the objective of representation learning has a general mathematical form that applies to many architectures and systems. This objective appropriately considers the uncertainty in choosing a correct representation. It represents a Bayesian framework for sensory processing and representation learning.

This chapter is adapted from a preprint[7] and is in the process of peer review.

[7] Benjamin, Ari S., and Konrad P. Kording. "Learning to infer in recurrent biological networks" arXiv preprint arXiv:2006.10811 (2021).

Abstract

A popular theory of perceptual processing holds that the brain learns a generative model of the world as well as a paired recognition model. Though this explains much of perception, it is not known how such models are learned. Seeking an algorithm that is compatible with the complex inter-dependencies of neurons induced by recurrence, we argue here that the cortex may learn with an adversarial algorithm. Many observable symptoms of this approach would resemble known neural phenomena, including wake/sleep cycles and oscillations that vary in magnitude with surprise. We describe how further predictions could be tested. We illustrate the idea on recurrent neural networks trained to model image and video datasets. This theory of learning brings variational inference closer to neuroscience and yields multiple testable hypotheses.

Introduction

Animals can predict future sensations, are surprised when predictions are violated, and learn from such surprises to predict better. These phenomena imply that animals construct internal models of the world, or at least of their sensations, to assess what is likely to occur. In this formalism, a surprising event corresponds to an event with low probability in one's internal model. Learning must adapt the internal model so that events are less surprising. Internal models carry clear benefits to a perceiving brain and are ubiquitous in computational neuroscience as theories of how the brain ought to perceive (Fiser et al., 2004; Wolpert et al., 1995).

Internal models predict must sensations by using internal representations of aspects of the world. However, these representations must be updated when new information is received. This requires interpreting the low-level sensations in the context of its internal model – the reverse direction of predicting sensations. In this framework, which is often attributed to Helmholtz, (Knill & Richards, 1996; von Helmholtz, 1925), perception is understood as a process of inferring what high-level factors in one's internal model are consistent with new observations. One may infer that a dark region is a shadow and not a coincidental dark spot, for example, because this is most consistent with one's internal model of the world. Many psychophysical and physiological experiments support this understanding of perception as inference, and as a result this idea is now central to the modern theory of perception (Berkes et al., 2011; Dasgupta et al., 2020; Friston, 2005; Hinton & Ghahramani, 1997; Kleinschmidt & Jaeger, 2015; Mumford, 1994; Poggio et al., 1987; Yuille & Kersten, 2006).

One of the ways in which the brain could perform inference over an internal model is via a 'recognition model' instantiated in bottom-up synapses (Dasgupta et al., 2020; Hinton et al., 1995). In this conception, the brain generates expectations over lower perceptual areas, such as V1, via top-down feedback. The bottom-up, feedfoward propagation of new information is then expected to be self-consistent with these expectations, in the sense that they invert the generative model. Learning a representation of the world becomes learning a good generative model while simultaneously learning invert it.

Here, we present a potential algorithmic interpretation of wake and sleep cycles that would allow organisms to learn internal models as well as a paired recognition model. Our model follows the pioneering paradigm of the Wake/Sleep algorithm (Hinton et al., 1995). However, we are motivated by the failure of this algorithm to model statistical dependencies between neurons (given a previous layer). Such dependencies are almost certainly present in the brain given the high degree of local recurrence. Reconciling the theoretical approach of the Wake/Sleep algorithm with the brain requires developing new algorithms compatible with its connectivity.

Our model and hypothesis introduces local aggregators of activity, i.e. interneurons, in order to solve this learning problem and allow Bayesian inference over neural populations. These cells take on the role of a teacher: they summarize the distributed activity of a target population in such a way as to indicate how incoming synapses should change. This requires these cells to themselves learn to be useful for learning, as a meta-learning rule.

The justification for this rule derives from the literature on adversarial learning (Goodfellow et al., 2014). The teacher-cell interneurons learn to classify, or discriminate, whether population activity is driven by a generative model or by a recognition model. Like the Wake-Sleep algorithm, this rule requires alternating between two phases, one driven by the generative model and one by the recognition model. Neurons that learn with this rule can effectively gate plasticity over connections such that they learn generative or recognition models.

As we demonstrate here, this algorithm is effective at learning hierarchical representations even when the neural populations are highly recurrent. In addition, we hypothesize a new role for a type of interneurons. We describe how this interpretation might be tested experimentally. This extension of the Wake/Sleep algorithm offers a bridge between the computational objective of Bayesian inference over an internal model and the biological implementation that makes learning possible.

Background: the goal of learning

Here we provide a brief restatement of the goal of learning. The objective of our learning problem is to learn mappings between inputs $\mathbf{x}$ and a learned representation $\mathbf{z}$ that are reciprocal in a specific sense. In our model both $\mathbf{x}$ and $\mathbf{z}$ are taken to be vectors of neural activities. This objective, sometimes called 'variational inference' or 'free energy minimization' is standard and can be found in various textbooks (Bishop, 2006; Dayan et al., 2001). We will introduce it here as the sum of two different objectives.

The first objective is to model the inputs. The top-down connections is to generate a probability distribution over $\mathbf{x}$ for each $\mathbf{z}$. This distribution can be written as $p_\theta(\mathbf{x}|\mathbf{z})$, and it depends on the parameters θ. Together with a prior distribution over $\mathbf{z}$, this implies a distribution over $\mathbf{x}$ generated by the model and its prior. This modeled distribution is $p_\theta(\mathbf{x})$. The parameters of the model θ are adjusted such that the true input distribution $q(\mathbf{x})$ equals $p_\theta(\mathbf{x})$.

The second objective is inference. The bottom-up network has the role of mapping an observed $\mathbf{x}$ to a distribution over $\mathbf{z}$, $q_\phi(\mathbf{z}|\mathbf{x})$. Note that this depends on the learnable parameters ϕ. Learning aims to adjust ϕ such that $q_\phi(\mathbf{z}|\mathbf{x})$ approximates the posterior distribution of the **z** could have generated an observed **x** under the internal model, i.e. $p_\theta(\mathbf{z}|\mathbf{x})$. The parameters ϕ are shared over all inputs. This setting is sometimes called 'amortized' variational inference.

When these objectives are taken together, it can be seen that the final objective is to match two probability distributions. Both of the above objectives are met when $q_\phi(\mathbf{x},\mathbf{z}) = p_\theta(\mathbf{x},\mathbf{z})$, i.e. when the joint probability distribution of *(real data, inferred representation)* pairs matches the joint distribution of *(sampled representation, generated data)* pairs. Learning in this framework is thus a distribution-matching problem.

One can match two distributions with many metrics. Most variational inference algorithms minimize the Kullbeck-Leibler (KL) divergence between these two joint distributions. The KL divergence between the two joint distributions is:

$$D_{KL}(q_\phi \| p_\theta) = \mathop{\mathbb{E}}_{q_\phi(\mathbf{x},\mathbf{z})}\left[\log \frac{q_\phi(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{x},\mathbf{z})}\right].$$

This expression is sometimes referred to as the variational free energy. Note that this is typically written using the conditional $q_\phi(\mathbf{z}|\mathbf{x})$, whereas we use the joint $q_\phi(\mathbf{x},\mathbf{z})$ to emphasize the distribution-matching interpretation. This does not change optimization because the value of the equation changes only by the entropy of the inputs **x**, which is independent of learnable parameters.

This objective is quite general and can describe probabilistic representation learning in many settings. Examples span from probabilistic principal components analysis to Gaussian mixture models to very complex hierarchical, nonlinear models. Note that the 'usefulness' of the representation **z**, however defined, depends on what prior and architecture is trained towards this objective. The objective itself is general to the form of the representation.

The problem posed by recurrence

Recurrence complicates learning because it prevents the use of a powerful simplification used by many algorithms. This is to treat each cell as being independent, given the upstream neurons. This allows learning to be local to each synapse. Specifically, the objective D_{KL} becomes a sum of single-neuron prediction errors and entropies. These can be computed locally (Rao & Ballard, 1999; Urbanczik & Senn, 2014). This strategy of connectivity restrictions underlies the bulk of variational inference algorithms proposed as models of the brain.

The effect of recurrence can be made precise by imagining one neuron A that predicts two neurons B and C which interconnect (Fig. 2a). In this setting, the objective of prediction (contained within D_{KL}) is to maximize the log probability of observing B and C given A, $log\ p(B,C|A)$. Recurrence between B and C introduces a dependency, meaning tha: $\log p\ (B,C|A) \neq \log p\ (B|A) + \log p\ (C|A)$. Thus B and C cannot be predicted separately. Synapses need access to all A,B, and C's activity in order to learn.

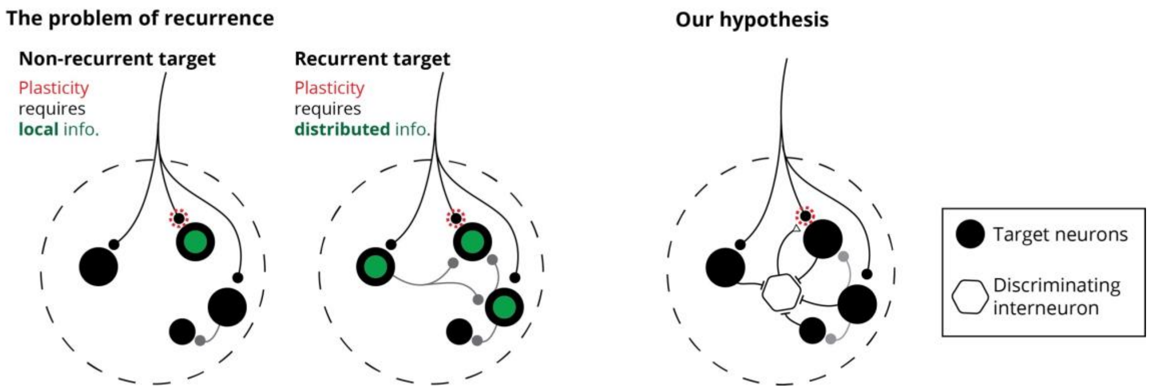

The problem of recurrence
Non-recurrent target
Plasticity requires local info.
Recurrent target
Plasticity requires distributed info.
Our hypothesis
Target neurons
Discriminating interneuron

Fig 1. Learning to predict a target population with fast recurrence. (left) The green neurons interconnect and are not independent given top-down activity; learning at the synapse highlighted in red requires information about all three. (right) This can be solved with an interneuron that aggregates local activity and signals how plasticity should change.

Ultimately, the problem posed by recurrence is that synapses now also require nonlocal information about many other neurons' activity. The solution is to find low-dimensional signals available for plasticity that *aggregate* local activity and communicate locally how learning should proceed (Fig. 1).

Results

Two adversarial algorithms for sensory learning

Here we describe two algorithms for sensory learning. The idea common to both algorithms is to leverage a switch between an externally-biased mode and an internally-biased mode of processing. Such switches are widely observed in neuroscience and it has been hypothesized that their purpose is to learn internal models (Honey et al., 2017). Our contribution is to propose specific algorithms that use such switches to learn representations and models of the world.

As reviewed in the introduction, the goal of representation learning is equivalent to aligning two probability distributions: the joint distributions of stimulus-driven activity $q_\phi(x,z)$ and the distribution of self-generated activity $p_\theta(x,z)$. To derive an algorithm that can align these distributions, we look to the literature on generative adversarial algorithms (Donahue et al., 2016; Dumoulin et al., 2016; Goodfellow et al., 2014). The insight of the adversarial approach is that two distributions can better aligned if a critic or discriminator can tell samples from the distributions apart.

The discriminator can be interpreted as a learned reward signal for sensory cortex. It signifies in which ways the two distributions meaningfully differ, and tricking the discriminator becomes the purpose of sensory learning. Importantly, this approach does not assume conditional independence and is compatible with local recurrence.

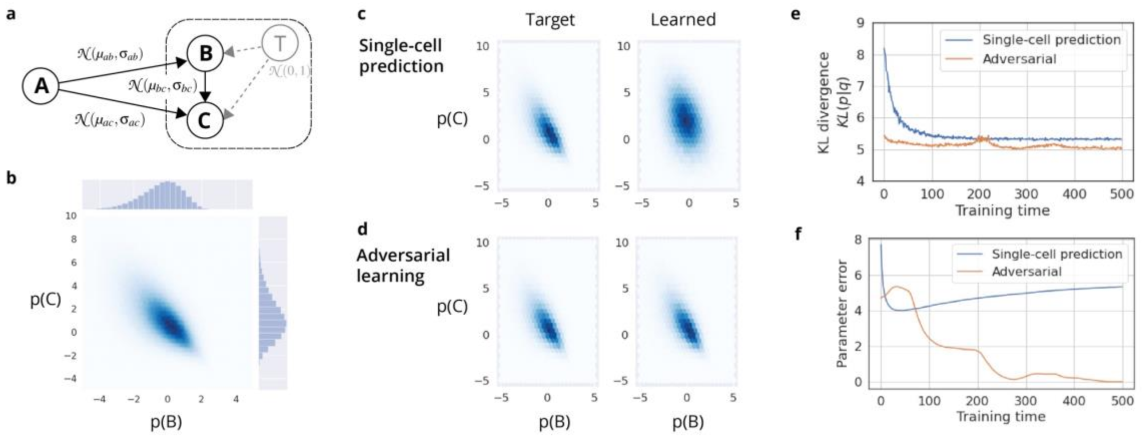


Fig 2. A simple case illustrating the problem of recurrence. a) We created a toy task of predicting two connecting neurons B and C from neuron A. All connections evoke Gaussian-distributed responses, and all neurons are linear. The "target population" is driven by hidden neuron T. b) The joint distribution $p(B,C)$ for a fixed $T \rightarrow B$ and $T \rightarrow C$ connections is quite correlated. c) When learning the $A \rightarrow B$ and $A \rightarrow C$ connections by maximizing $p(B|A)$ and $p(C|A)$ separately, ignoring recurrence, the predictions fail to match the joint distribution. Left is repeated from (b), right is $E_A[p(B,C|A)]$. d) The adversarial hypothesis (involving a new discriminator interneuron, not shown) successfully aligns the distribution. The discriminator sees B and C and gates plasticity at the predictive connections from A. e) The alignment can be quantified with the KL divergence (calculated via hexagonal histograms) between the learned and target distribution. f) Another measure of success is the distance between A and T's outward connections' mean and variance, which we call the parameter error.

Comparing wake and sleep distributions

It has recently been argued that cortical activity during sleep indeed represent samples from an internal model (Aru et al., 2020; Hobson & Friston, 2012; Honey et al., 2017). Much of the evidence for this is phenomenological; dreams, after all, have many of the same statistical properties of the world such as meaningful objects and sounds. This idea has also long been popular in computational theory (Ackley et al., 1985; Hinton et al., 1995). If this is indeed the case, it would provide a powerful opportunity to the brain to use an adversarial algorithm for learning representations.

What is required for a neuron to act as a discriminator is a particular form of plasticity that changes sign with the phase of mode-switching. One simple method is to attempt to have a high firing rate on samples from one distribution and low firing rate on the other.

If we represent the discriminator as a neuron with an output given by $D(\mathbf{x},\mathbf{z})$, which is a function of the entire network state, it should have plasticity that maximizes the objective:

$$\mathcal{L}_{WS} = \mathop{\mathbb{E}}_{q_\phi(\mathbf{x},\mathbf{z})}\Big[D(\mathbf{x},\mathbf{z})\Big] - \mathop{\mathbb{E}}_{p_\phi(\mathbf{x},\mathbf{z})}\Big[D(\mathbf{x},\mathbf{z})\Big]$$

The particular formulation of Eq. 2, the difference of the average of D between phases, is but one among a large number of adversarial objectives suitable for matching distributions (Nowozin et al., 2016). It is that of the Wasserstein GAN, which can be derived from a minimization of the Wasserstein-1 distance between probability distributions (Arjovsky et al., 2017) and requires an additional regularizing penalty so that $\nabla_{x,z}D(\mathbf{x},\mathbf{z}) \leq 1$. As proved by Donahue et al. (2016) and Dumoulin et al. (2016), this strategy converges when the recognition model and generative model are inverses.

This strategy is very effective for aligning distributions. In Figure 2, we show a simple demonstration of this algorithm. The task is designed to show the minimal case in which local recurrence causes problems. This when a source neuron (A) attempts to predict to two neurons in a target population (B and C), but these neurons interconnect. (Note that for this demonstration we consider only half of the problem, learning the internal model of B and C given A, and omit the problem of inverting this model, inferring A given B and C). This causes dependencies in their distribution given the source neuron A (Fig. 2b). Learning an internal model of B and C by predicting them separately causes poor learning (Fig. 2c). Thus B and C cannot be predicted separately; synapses need access to all A,B, and C's activity. However, the adversarial strategy allows the synapses from the source neuron A to correctly model their distribution (Fig. 2d).

A purely wake/sleep discriminator receiving input from all neurons would work well in small nervous systems. This is because the discriminator has an input dimension equal to the total number of neurons. In the ML literature, applications of this algorithm have been limited to dimensions of **x** and **z** totalling tens of thousands of units at the largest (Donahue & Simonyan, 2019). Scaling to a system as large as the neocortex motivates our second proposal.

Comparing phases of an ascending oscillation

The cortex arguably switches between modes of processing on multiple timescales. In addition to wake and sleep, oscillations with periods as low as 10ms have also been hypothesized to represent a switch of the main drive of activity from external (bottom-up) to internal (top-down) sources (Honey et al., 2017). Such oscillations may also be used for adversarial learning. This allows for local discriminators with many fewer inputs than in a purely Wake/Sleep algorithm.

An algorithm can be derived by examining the KL divergence $D_{KL}(q\|p)$ for a multilayer network with internal recurrence in each layer. In this architecture, we can write the joint log-probabilities as:

$$\log q_\phi(\mathbf{x},\mathbf{z}) = \sum_i \log q_\phi(\mathbf{z}_i|\mathbf{pa}_i^q) \qquad \Big| \qquad \log p_\theta(\mathbf{x},\mathbf{z}) = \sum_i \log p_\theta(\mathbf{z}_i|\mathbf{pa}_i^p)$$

Here $\mathbf{pa}_i^p$ represent the neurons presynaptic to the population xi in the generative phase, and $\mathbf{pa}_i^q$ are the neurons presynaptic in the inference phase. These terms are the KL divergence between the generative distribution and inference distribution over zi:

$$\mathop{\mathbb{E}}_{q_\phi(\mathbf{x},\mathbf{z})}\left[\log \frac{q_\phi(\mathbf{z}_i|\mathbf{z}_{i-1})}{p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1})}\right] = \mathop{\mathbb{E}}_{\mathbf{z}_{i-1},\mathbf{z}_{i+1}\sim q_\phi(\mathbf{x},\mathbf{z})}\left[KL(q_\phi(\mathbf{z}_i|\mathbf{z}_{i-1})\|p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1}))\right]$$

If these layerwise KL divergences can be minimized, the overall $D_{KL}(q_\phi\|p_\theta)$ will decrease as well. Instead of using one discriminator that sees the entire network state, then, the brain could use one discriminator per layer that only observes the local population.

In the W-GAN formalism, the objective maximized by the discriminator and minimized by the layer would be:

$$\mathcal{L}_{O,\mathbf{z}_i} = \mathop{\mathbb{E}}_{q_\phi(\mathbf{x},\mathbf{z})}\Big[D_i(\mathbf{z}_i)\Big] - \mathop{\mathbb{E}}_{q_\phi(\mathbf{z}_{i+1}|\mathbf{x})}\ \mathop{\mathbb{E}}_{p_\phi(\mathbf{z}_i|\mathbf{z}_{i+1})}\Big[D_i(\mathbf{z}_i)\Big]$$

Evaluating this objective requires obtaining samples of activity in each phase. While the inference distribution is straightforward to sample, the generative distribution $p_\theta(z_i|z_{i+1})$, $z_{i+1} \sim q_\phi$ requires an oscillation in which the neurons zi are driven by feedback from one layer up. Thus the discriminator would observe alternating phases of somatic activity, with one phase corresponding to bottom-up inputs and one phase corresponding the top-down prediction.

If oscillations represent mode switches, then one should observe stronger oscillations in activity for unexpected stimuli. After all, for expected (i.e. well-modeled) stimuli the top-down predictions are indistinguishable from bottom-up signals. We thus expect to see the power of the oscillation correlate with surprise, as is observed for the gamma oscillations. For example, in auditory cortex, MEG and EEG studies show increases in gamma power to unexpected auditory stimuli (Haenschel et al., 2000; Todorovic et al., 2011), to omissions of expected musical beats (Fujioka et al., 2009), and to unexpected mismatches between auditory and visual cues (Arnal et al., 2011). In the hippocampus, both theta and gamma ranges increase after unexpected stimuli (Axmacher et al., 2010). The correlation of prediction errors with oscillation strength is consistent with the overall framework.

The adversarial interpretation of oscillations makes two further predictions. First, the oscillations should ascend up sensory hierarchies. Indeed, oscillations in the gamma range have been found to ascend up the visual hierarchy (Van Kerkoerle et al., 2014). Second, top-down feedback in the internal mode should drive somatic activity. Other algorithms for Bayesian inference require that feedback only be integrated in dendritic compartments for comparison with bottom-up activity (Friston, 2005; Siegel et al., 2000). In reality, feedback into apical dendrites can have a large effect on somatic activity (Larkum et al., 1999).

Learning on two timescales: Learning can occur on both timescales, and with both algorithms. While one could define a separate discriminator each algorithm, it works well to combine objectives on the same set of discriminators. Defining a hyperparameter γ that balances between the objectives, we obtain a combined minimax objective for the ith layer and its discriminator: $L_{\mathbf{z}_i} = \gamma L_{WS} + (1 - \gamma)(L_{O,\mathbf{z}_i})$.

Experiments

To illustrate our proposal, here we apply the learning algorithm presented above. The overall framework of adversarially learned inference over a generative model has been validated elsewhere in the context of feedforward networks (Belghazi et al., 2018; Donahue et al., 2016; Dumoulin et al., 2016; Larsen et al., 2016; Pu et al., 2017; Srivastava et al., 2017). As a crucial test of biological feasibility, here we will empirically ask how well the algorithm will work on recurrent neural network architectures.

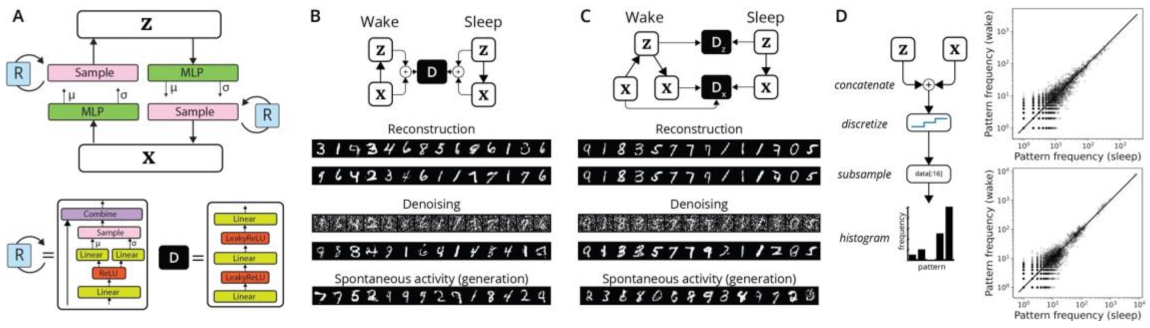


Fig 3. Generating MNIST digits with a recurrent, stochastic autoencoder. A) In either inference or generation, samples are passed through a multilayer network parameterizing a Gaussian, which is then sampled and subject to nonlinear stochastic recurrence. B) In the adversarial wake/sleep algorithm (Section 3.1), a global discriminator observes both z and x during wake and sleep phases. This algorithm allows realistic generation but often shows poor reconstruction. C) When the discriminator aligns inference with layer-wise reconstructions (Section 3.2), discriminators are local to each layer. We trained this system with the combined objective of Eq. 6 with $\gamma = 0.1$. D) As in Berkes et al. (2011), we can check empirically if the joint distributions of the Sleep and Wake phases match. After quantizing the vector of x and z by rounding to the nearest integer, we observed patterns over 16 units' activity and histograms of how often these patterns appeared when observing/generating MNIST digits. Comparisons are shown for the model in panel B (top) and in panel C (bottom).

MNIST digit generation from stochastic recurrent representations

We begin with a simple autoencoder in which both the recognition model (encoder) and internal model of inputs (decoder) are stochastic, nonlinear transformations followed by recurrence (Fig. 3). The first-stage transformations are 2-layer fully-connected neural networks that output the mean and variance of a diagonal Gaussian over the latent z or inputs x, which is then sampled. During recurrence this sample is given to a stochastic network of the same architecture, the output of which is merged with the original sample via an arithmetic mean. As a result of the nonlinearity and interpolation, the conditional distributions $q\phi(z|x)$ and $p\theta(x|z)$ are not Gaussian. This recurrence induces dependencies among neurons in each layer, given the previous layer.

We found that recurrence exacerbates a problem with the purely Wake/Sleep approach. While generation is of good quality, reconstructing digits produces different digits than those input (Fig. 3B). Reconstruction maps to the manifold of MNIST digits, but elsewhere. This is despite the optimal solution being perfect inversion. A wide search over learning rates, weight decay, and the β parameters of the Adam optimizer did not produce better reconstructions.

The oscillatory algorithm was much more stable (Fig. 3C). In addition to the basic adversarial loss (LO, Eq. 5), we also employed the approach of the VAE-GAN of using the hidden layers of the discriminator as a metric between inputs and reconstructions (Larsen et al., 2016). This objective resulted in good reconstructions, signifying that the inference network maps to the support of the generative posterior.

To measure the success beyond visual inspection, we employed the approach of Berkes et al. (2011) and empirically quantified the overlap of the wake and sleep joint distributions. After quantizing the inputs and latents to the nearest integer and subsampling to 16 units (8 x and 8 z), we counted how many times each unique pattern over the 16 units appeared in wake or sleep. This approximate approach is necessary as the log-likelihood is not tractable due to recurrence. In Fig. 3D it can be seen that both algorithms are sufficient to align the joint distributions as closely as in Berkes et al. (2011).

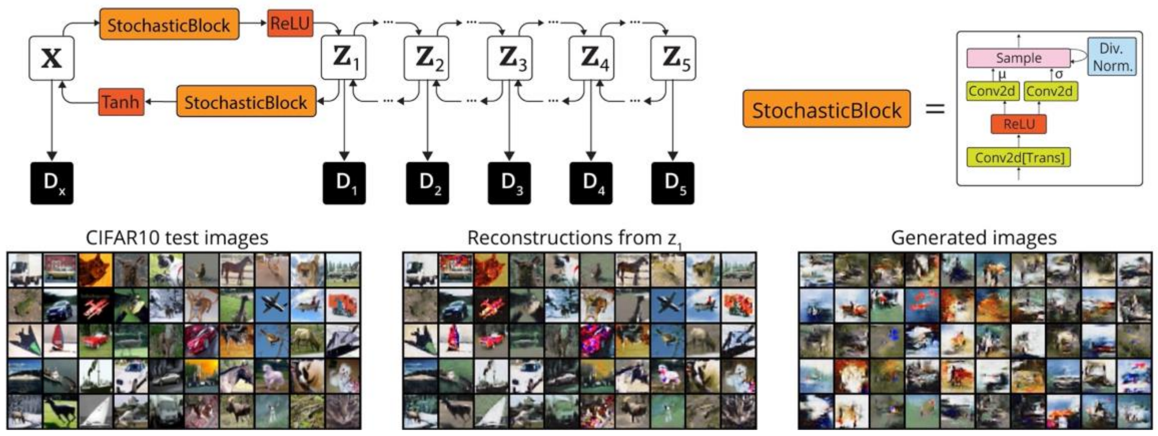


Fig 4. Learning to model images from CIFAR-10 with a hierarchical architecture and convolutional filters. Each layer is a stochastic convolutional neural network subject to divisive normalization. We trained towards Eq. 6 with $\gamma = 0.5$.

Modeling CIFAR-10 with a hierarchical model

To test hierarchical models, we trained a stochastic variation of the DCGAN architecture using the combined objective of Eq. 6 (Fig. 4). Each of the five layers of latent variables and the input is paired with its own discriminator. The conditional distribution of each layer is a reparameterized diagonal Gaussian, but passed through a ReLU nonlinearity and then subject to divisive normalization over each spatial location, a ubiquitous form of local recurrence in the cortex (Heeger, 1992) that is also known to stabilize GAN training (Karras et al., 2017). We trained with the combined objective with $\gamma = 0.5$; additional model and training details can be found in the Appendix. After 400 epochs, inference again maps to the support of the generative posterior as evidenced by reconstructions (Fig. 4B). Thus, a purely adversarial approach is feasible for deeper networks and more naturalistic datasets.

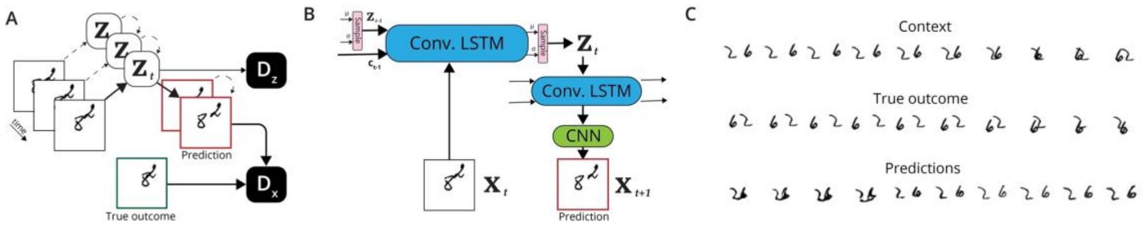


Fig 5. A) We trained a recurrent, autoencoding architecture to predict future frames in the Moving MNIST task. Discriminators align inputs and hidden state distributions. B) Encoding and decoding architecture. Hidden states transfer stochastically. C)Conditioned on ten context frames, the trained network generates plausible digits and trajectories.

Predicting future inputs

Recurrence has a clear benefit when inputs vary in time, and as a model of brain processing the temporal dimension is unavoidable. Our approach can readily apply to this setting. Because of conduction delays in real neurons, however, the goal of a loop of processing changes from autoencoding to predicting the future inputs.

Adversarial approaches to video prediction are common in the generative literature. Directly applying the oscillatory algorithm to video prediction, in fact, nearly produces the algorithm proposed in Stochastic Adversarial Video Prediction (SAVP; Lee et al. (2018)). Like the VAE-GAN from which it derives, SAVP regularizes $q_\phi(\mathbf{z}_t)$ to the prior $p(\mathbf{z})$ by approximating it as a diagonal Gaussian, and additionally applies an L_1 loss over $\mathbf{x}_t$. Both imply the assumption that the conditional distributions $q_\phi(\mathbf{z}_t|\mathbf{x}_{0:t})$ and $p_\theta(\mathbf{x}_{t+1}|\mathbf{z}_{0:t})$ can factorize into single-neuron terms. Here, we demonstrate that the algorithm works when all training is adversarial and the distributions are allowed to be non-Gaussian.

We trained a stochastic architecture (Fig. 5B) to predict the future frames of video in the Moving MNIST task, in which two digits bounce for 20 frames in a 64x64 area with random initial velocities (Srivastava et al., 2015). Both the encoding and decoding networks are convolutional LSTMs with stochastic hidden state transitions.

We used separate discriminators for **x** and **z** to align the joint distribution of latent vectors and subsequent inputs, $(\mathbf{x}_t, \mathbf{z}_{t-1})$ via Eq. 6. The similarity between the true outcome $\mathbf{x}_t$ and the prediction $\hat{\mathbf{x}}_t$ were also minimized using the hidden layers of the discriminator as a metric. After training, the encoding and decoding are able to predict plausible future video frames (Fig. 5C).

Discussion

We have proposed that an adversarial algorithm could provide the missing link between a widely hypothesized computational goal of learning and its implementation in neural systems. This provides a concrete implementation of the idea that the brain learns internal models by switching between externally- and internally- driven modes of processing (Honey et al., 2017). Our experiments demonstrate that this algorithm is compatible with the brain's stochasticity, recurrence, and multilevel architecture.

A central prediction is that the brain meta-learns an objective for sensory learning in the form of a discriminator. This discriminator learns by trying to increase activity during a stage of sleep and decrease activity during wake, or alternatively in the two phases of a fast oscillation. This approximate objective is arguably easier to construct for biological areas than directly estimating the variational free energy of the entire sensory cortex, as would be required if meta-learning were not used (Rezende & Gerstner, 2014).

What would the a discriminator look like? In the neocortex, we hypothesize local discriminators in each cortical column that classify oscillations. Due to their local connectivity these would resemble interneurons with control over local plasticity, especially in the critical period. Somatostatin-positive interneurons meet this rather broad criterion (Yaeger et al., 2019), as do 5-$HT_{3A}R^{+}$ cells in Layer 1 (Takesian et al., 2018) and likely many others.

The learning rules implied by our algorithm are perhaps the most distinguishing feature of a discriminator. Connections onto local discriminatory interneurons should have a plasticity rule that switches polarity with the phase of processing. Likewise, the way in which their activity affects plasticity in surrounding cortex should switch polarity depending on the phase of processing. There is evidence for phasic switches in plasticity in the brain. In the hippocampus, for example, transitions between potentiation and depression and have been observed with phases of the theta rhythm in hippocampus (Huerta & Lisman, 1995; Hyman et al., 2003; Pavlides et al., 1988). Synaptic strengths across cortex are also known to homeostatically increase during wake and decrease during sleep (González-Rueda et al., 2018; Hengen et al., 2016; Pacheco et al., 2021). Our approach offers a algorithmic interpretation of such phasic switches of plasticity.

Possible experiments

While phenomena such as hippocampal replay show that activity in wake and sleep are closely related (Nádasdy et al., 1999), little is known about the statistical distributions of brain activity during sleep. A study investigating this could closely resemble Berkes et al. (2011), which compared the distribution of activity in the ferret visual cortex when eyes are open and closed. Instead of comparing spontaneous closed-eye activity, one could compare activity during phases of LFP oscillations or during sleep. These distributions should align over development. A positive outcome in this experiment would extend indirect evidence that these internally-biased phases produce samples from an internal model (Hobson & Friston, 2012; Honey et al., 2017).

Perturbations would allow a deeper test. During the critical period of sensory learning, one could selectively silence activity during a stage of sleep or half of an oscillatory cycle, perhaps via optogenetic methods (Andrasfalvy et al., 2010). This perturbs the generative distribution. One could then observe if activity changes in the waking or opposing phase to match that perturbed distribution. A search for what mediates this alignment could provide a mechanism for adversarial sensory learning.

Recent advances in transcriptomic techniques may also help unveil the cell-specific roles that neuron types play in learning (Bernard et al., 2012). The hypothesis that a cell plays the role of a wake/sleep discriminator should be testable with e.g. Cre-dependent single-cell RNA-seq due to the slow timescale of the plasticity switch. A survey of interneuronal Cre lines should produce a cell type whose expression of factors relating to postsynaptic LTP and LTD is dominated by the sleep cycle.

Limitations

As a model of sensory learning, our hypothesis inherits the limitations of the broader computational goal. The strategy of Bayesian inference over a generative model is under-specified as a complete learning objective because it makes no reference to what subsets of information should be represented by higher sensory areas, or in what form. Learning good representations requires priors over appropriate responses or learned cognitive goals, perhaps embedded in the architecture. The algorithm we propose must act over a supplied architecture and set of priors.

While we have advocated here for variational Bayesian inference over vectors of neural activity, it is worth mentioning that the brain may perform Bayesian inference through other strategies such as particle filtering (Lee & Mumford, 2003) or loopy belief propagation (Raju & Pitkow, 2016). The brain may also infer and generate probability distributions implied by the activity of neurons (Vértes & Sahani, 2018).

We assume that the brain contains systems for efficiently minimizing objectives represented in neural activity. In our implementation, all learning depended on the backpropagation of error. Other systems for error minimization may be sufficient for this task. Alternatively, due to a connection between backpropagation and variational autoencoders, there is the possibility that the learned feedforward and feedback connections could themselves could be used for the credit assignment problem (Bengio, 2014; Lillicrap et al., 2020).

Methods

Software and hardware

The experiments presented were coded in Pytorch v1.8. All experiments were run on NVIDIA GeForce GTX 1080Ti GPUs. Running on a single card, the experiments take about 5 minutes to train the MNIST architecture, 4 hours to train the CIFAR-10 architecture, and 10 hours to train the Moving MNIST architecture. Searches over training details were performed most for the MNIST and CIFAR-10 tasks and required around 100 times these totals.

Optimization

For all tasks we used the Adam optimizer Kingma & Ba (2014) with $\beta_1 = 0.5$, $\beta_2 = 0.99$, and weight decay of 2×10^{-5}. The batch size was 512 except for the Moving MNIST task, which for memory reasons was 32. The learning rate of the inference and generation networks was 10^{-4} and the discriminator's learning rate of 4×10^{-4}. These learning rates decreased by a factor of 0.96 between epochs 200 and 250. The total number of training epochs for all tasks was 400. These hyperparameters were chosen from previous literature (specifically (Donahue et al., 2016)). Random hyperparameter searches were also conducted, but these did not overly improve results. In general we found that adversarial training is very fragile to the balance of architecture, learning rate, and optimizer between the generator and discriminator. Architectures were tuned by hand by visually examining the quality of reconstructions on the training set and generations from random noise. For all adversarial objectives we used the Wasserstein-GAN (Arjovsky et al., 2017) with a gradient penalty of $\lambda = 1$ (Gulrajani et al., 2017).

MNIST architecture

For the MNIST task (Bottou et al., 1994) in Figure 3, we used the train set for training and display reconstructions using the test set. MNIST is licensed CC BY-SA 3.0. Before training and generation, we rescaled inputs to the range $[-1,1]$. The architecture of both the inference and generation networks is a fully-connected, two-layer neural network with 512 hidden units and 128 output units (or 784^2 in the case of generation). The hidden nonlinearities are ReLUs. Half of the output units specify the standard deviation and half the mean of a diagonal Gaussian over the outputs. This is then sampled and fed to nonlinear recurrence with the same architecture as the inference/generation networks. Finally, the output is recombined via an arithmetic mean with the inputs.

In the global discriminator setup of Figure 3B, the discriminator first concatenated **x** and **z**, then passed these through a 3-layer fully-connected neural network with linear outputs. Each layer had 512 hidden units and a LeakyReLU unit with negative slope 0.2. In the local discriminator setup of Figure 3C, each discriminator for **x** and **z** was also a 3-layer network with 512 hidden units and LeakyReLU units.

CIFAR-10 architecture

For the CIFAR-10 task (Krizhevsky et al., 2009), available under an MIT licence, we trained an architecture inspired by the DCGAN (Radford et al., 2015). This dataset does not contain faces or other personal identifying information.

Our architecture contains a hierarchy of 5 layers of latent variables. The first layer above the inputs has 32 channels, which doubles every layer, except for the highest layer which has 100 channels. This is like the DCGAN architecture, but in ours both inference and generation each layer is stochastic. To allow backpropagation through stochasticity, each layer is a reparameterized Gaussian (Kingma & Welling, 2013) subject to divisive normalization applied over each pixel dimension (i.e. across channels) as in Karras et al. (2017). The the mean and standard deviation are given by a convolutional network with one hidden ReLU layer, linear outputs for the mean, and softplus outputs for the standard deviation. In order for the inference and generator networks to have the same architecture, the generator must use transposed convolutions where the inference uses convolutional operators.

Each of the 5 layers as well as the input are paired with their own discriminator. The 6 discriminators are similar in that they are all neural networks with two convolutional layers followed by a linear output, LeakyReLU activations (0.2 negative slope), and pixel-wise divisive normalization on the hidden layers. They differ only in the strides, kernel, and padding of the convolutional layers, which are decreased higher in the network to accommodate the shrinking spatial dimension. For further details about stride and padding, consult the accompanying code.

Moving MNIST architecture

Moving MNIST is available under an MIT license (Srivastava et al., 2015). Our architecture is inspired by Lee et al. (2018) in that encoding and decoding both involve convolutional LSTMs (Shi et al., 2015). We modified the standard deterministic LSTMs such that the output parameterizes a diagonal Gaussian over the hidden state, which is sampled each time the state updates. The stochasticity and nonlinearity implies that over time the distribution of hidden states given the same history of inputs is complex and not

Gaussian. Our model is one layer deep, and thus has one stochastic hidden state we perform approximate inference over.

The encoding and decoding LSTMs each have 32 hidden units, a kernel size of 3, padding of 1, and stride of 1. This preserves the spatial dimension. The decoding pathway additionally has a two-layer CNN to map the hidden state of the decoding LSTM to the inputs. This CNN has one hidden layer with 16 channels and a ReLU activation.

The discriminator used in this task is a similar architecture as in the CIFAR-10 task. Two convolutional layers with batch normalization and LeakyReLU units are followed by a linear layer, and the features before the linear layer are used as a metric to compare predictions with true outcomes. Note that the discriminator is not recurrent; each frame is compared separately.

Training proceeds as follows. The encoder and decoder LSTMs are first initialized by encoding and decoding the first 10 "context" frames. During this time period nothing is done with the prediction of the decoder, but hidden states are maintained. Gradients are not cut off to allow backpropagation through time for the future objective. In the second 10 frames, the discriminator over $\mathbf{x}$ attempts to classify predictions $\hat{\mathbf{x}}_t$ from true outcomes $\mathbf{x}_t$. The encoder and decoder pathways attempt to trick this, which due to the dependency on previous states requires backpropagation through time. Additionally, the encoder and decoder attempt to reduce the L_2 distance of the discriminator's penultimate layer given $\hat{\mathbf{x}}_t$ and $\mathbf{x}_t$. These two objectives ensure that the predictions of the next frame are both of a realistic style and appropriately similar to the true evolution of outcomes over time.

This solely discriminator-guided prediction method worked well. Interestingly, this places no constraints over $\mathbf{z}$. However, for consistency with the rest of the manuscript, we also trained the discriminator with a sleep phase, enforcing that the distribution over $\mathbf{z}$ matches a prior. The sleep phase involved sampling the hidden state from a prior distribution over $\mathbf{z}$ (a standard normal), and then bouncing back and forth between $\mathbf{x}$ and $\mathbf{z}$ over 10 timesteps using the encoder and decoder pathways. This wake/sleep objective is discounted by $\gamma = 0.5$ in the same manner as Eq. 6.

The Pytorch code used for all figures in this manuscript is available at https://github.com/KordingLab/adversarial-wake-sleep.

References

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

Bertalan K Andrasfalvy, Boris V Zemelman, Jianyong Tang, and Alipasha Vaziri. Two-photon single-cell optogenetic control of neuronal activity by sculpted light. *Proceedings of the National Academy of Sciences*, 107(26):11981–11986, 2010.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Luc H Arnal, Valentin Wyart, and Anne-Lise Giraud. Transitions in neural oscillations reflect prediction errors generated in audiovisual speech. *Nature neuroscience*, 14(6): 797, 2011.

Jaan Aru, Francesca Siclari, William A Phillips, and Johan F Storm. Apical drive—a cellular mechanism of dreaming? *Neuroscience & Biobehavioral Reviews*, 2020.

Nikolai Axmacher, Michael X Cohen, Juergen Fell, Sven Haupt, Matthias Dümpelmann, Christian E Elger, Thomas E Schlaepfer, Doris Lenartz, Volker Sturm, and Charan Ranganath. Intracranial eeg correlates of expectancy and memory formation in the human hippocampus and nucleus accumbens. *Neuron*, 65(4):541–549, 2010.

Mohamed Ishmael Belghazi, Sai Rajeswar, Olivier Mastropietro, Negar Rostamzadeh, Jovana Mitrovic, and Aaron Courville. Hierarchical adversarially learned inference. *arXiv preprint arXiv:1802.01071*, 2018.

Yoshua Bengio. How auto-encoders could provide credit assignment in deep networks via target propagation. *arXiv preprint arXiv:1407.7906*, 2014.

Pietro Berkes, Gergo Orbán, Máté Lengyel, and József Fiser. Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. *Science*, 331(6013):83–87, 2011. ISSN 00368075. doi: 10.1126/science.1195870.

Amy Bernard, Laura S Lubbers, Keith Q Tanis, Rui Luo, Alexei A Podtelezhnikov, Eva M Finney, Mollie ME McWhorter, Kyle Serikawa, Tracy Lemon, Rebecca Morgan, et al. Transcriptional architecture of the primate neocortex. *Neuron*, 73(6): 1083–1099, 2012.

Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Larry D Jackel, Yann LeCun, Urs A Muller, Edward Sackinger, Patrice Simard, et al. Comparison of classifier methods: a case study in handwritten digit recognition. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5)*, volume 2, pp. 77–82. IEEE, 1994.

Ishita Dasgupta, Eric Schulz, Joshua B Tenenbaum, and Samuel J Gershman. A theory of learning to infer. *Psychological Review*, 127(3):412, 2020.

Peter Dayan, Laurence F Abbott, et al. *Theoretical neuroscience*, volume 806. Cambridge, MA: MIT Press, 2001.

Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In *Advances in Neural Information Processing Systems*, pp. 10541–10551, 2019.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

József Fiser, Chiayu Chiu, and Michael Weliky. Small modulation of ongoing cortical dynamics by sensory input during natural vision. *Nature*, 431(7008):573–578, 2004. ISSN 00280836. doi: 10.1038/nature02907.

Karl Friston. A theory of cortical responses. *Philosophical transactions of the Royal Society B: Biological sciences*, 360(1456):815–836, 2005.

Takako Fujioka, Laurelj Trainor, Edwardw Large, and Bernhard Ross. Beta and gamma rhythms in human auditory cortex during musical beat processing. *Annals of the New York Academy of Sciences*, 1169(1):89–92, 2009.

Ana González-Rueda, Victor Pedrosa, Rachael C Feord, Claudia Clopath, and Ole Paulsen. Activity-dependent downscaling of subthreshold synaptic inputs during slow-wave-sleep-like activity in vivo. *Neuron*, 97(6):1244–1252, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pp. 5767–5777, 2017.

Corinna Haenschel, Torsten Baldeweg, Rodney J Croft, Miles Whittington, and John Gruzelier. Gamma and beta frequency oscillations in response to novel auditory stimuli: a comparison of human electroencephalogram (eeg) data with in vitro models. *Proceedings of the National Academy of Sciences*, 97(13):7645–7650, 2000.

David J Heeger. Normalization of cell responses in cat striate cortex. *Visual neuroscience*, 9(2):181–197, 1992.

Keith B Hengen, Alejandro Torrado Pacheco, James N McGregor, Stephen D Van Hooser, and Gina G Turrigiano. Neuronal firing rate homeostasis is inhibited by sleep and promoted by wake. *Cell*, 165(1):180–191, 2016.

Geoffrey E Hinton and Zoubin Ghahramani. Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 352(1358):1177–1190, 1997.

Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214): 1158–1161, 1995.

J Allan Hobson and Karl J Friston. Waking and dreaming consciousness: neurobiological and functional considerations. *Progress in neurobiology*, 98(1):82–98, 2012.

Christopher J Honey, Ehren L Newman, and Anna C Schapiro. Switching between internal and external modes: a multiscale learning principle. *Network Neuroscience*, 1 (4):339–356, 2017.

Patricio T Huerta and John E Lisman. Bidirectional synaptic plasticity induced by a single burst during cholinergic theta oscillation in ca1 in vitro. *Neuron*, 15(5): 1053–1063, 1995.

James M Hyman, Bradley P Wyble, Vikas Goyal, Christina A Rossi, and Michael E Hasselmo. Stimulation in hippocampal region ca1 in behaving rats yields long-term potentiation when delivered to the peak of theta and long-term depression when delivered to the trough. *Journal of Neuroscience*, 23(37):11725–11731, 2003.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Dave F Kleinschmidt and T Florian Jaeger. Robust speech perception: recognize the familiar, generalize to the similar, and adapt to the novel. *Psychological review*, 122 (2):148, 2015.

David C Knill and Whitman Richards. *Perception as Bayesian inference*. Cambridge University Press, 1996.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Matthew E Larkum, J Julius Zhu, and Bert Sakmann. A new cellular mechanism for coupling inputs arriving at different cortical layers. *Nature*, 398(6725):338–341, 1999.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pp. 1558–1566. PMLR, 2016.

Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018.

Tai Sing Lee and David Mumford. Hierarchical bayesian inference in the visual cortex. *JOSA A*, 20(7):1434–1448, 2003.

Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

David Mumford. Neuronal architectures for pattern-theoretic problems. *Large-scale neuronal theories of the brain*, pp. 125–152, 1994.

Zoltán Nádasdy, Hajime Hirase, András Czurkó, Jozsef Csicsvari, and György Buzsáki.
Replay and time compression of recurring spike sequences in the hippocampus. *Journal of Neuroscience*, 19(21):9497–9507, 1999.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *arXiv preprint arXiv:1606.00709*, 2016.

Alejandro Torrado Pacheco, Juliet Bottorff, Ya Gao, and Gina G Turrigiano. Sleep promotes downward firing rate homeostasis. *Neuron*, 109(3):530–544, 2021.

Constantine Pavlides, Yoram J Greenstein, Mark Grudman, and Jonathan Winson. Long-term potentiation in the dentate gyrus is induced preferentially on the positive phase of θ-rhythm. *Brain research*, 439(1-2):383–387, 1988.

Tomaso Poggio, Vincent Torre, and Christof Koch. Computational vision and regularization theory. *Readings in computer vision*, pp. 638–643, 1987.

Yuchen Pu, Weiyao Wang, Ricardo Henao, Liqun Chen, Zhe Gan, Chunyuan Li, and Lawrence Carin. Adversarial symmetric variational autoencoder. In *Advances in neural information processing systems*, pp. 4330–4339, 2017.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Rajkumar Vasudeva Raju and Xaq Pitkow. Inference by reparameterization in neural population codes. *arXiv preprint arXiv:1605.06544*, 2016.

Rajesh PN Rao and Dana H Ballard. Hierarchical Predictive Coding Model Hierarchical Predictive Coding of Natural Images. *Nature Neuroscience*, 2(1):79–87, 1999. URL http://neurosci.nature.com.

Danilo Rezende and Wulfram Gerstner. Stochastic variational learning in recurrent spiking networks. *Frontiers in Computational Neuroscience*, 8:38, 2014. ISSN 1662-5188. doi: 10.3389/fncom.2014.00038. URL https://www.frontiersin.org/article/10.3389/fncom.2014.00038.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *arXiv preprint arXiv:1506.04214*, 2015.

Markus Siegel, Konrad P Körding, and Peter König. Integrating top-down and bottom-up sensory processing by somato-dendritic interactions. *Journal of computational neuroscience*, 8(2):161–173, 2000.

Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pp. 3308–3318, 2017.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pp. 843–852. PMLR, 2015.

Anne E Takesian, Luke J Bogart, Jeff W Lichtman, and Takao K Hensch. Inhibitory circuit gating of auditory critical-period plasticity. *Nature neuroscience*, 21(2): 218–227, 2018.

Ana Todorovic, Freek van Ede, Eric Maris, and Floris P de Lange. Prior expectation mediates neural adaptation to repeated sounds in the auditory cortex: an meg study. *Journal of Neuroscience*, 31(25):9118–9123, 2011.

Robert Urbanczik and Walter Senn. Learning by the dendritic prediction of somatic spiking. *Neuron*, 81(3):521–528, 2014.

Timo Van Kerkoerle, Matthew W Self, Bruno Dagnino, Marie-Alice Gariel-Mathis, Jasper Poort, Chris Van Der Togt, and Pieter R Roelfsema. Alpha and gamma oscillations characterize feedback and feedforward processing in monkey visual cortex. *Proceedings of the National Academy of Sciences*, 111(40):14332–14341, 2014.

Eszter Vértes and Maneesh Sahani. Flexible and accurate inference and learning for deep generative models. *arXiv preprint arXiv:1805.11051*, 2018.

Hermann von Helmholtz. *Treatise on Physiological Optics Volume III*. The Optical Society of America, 1925.

Daniel M Wolpert, Zoubin Ghahramani, and Michael I Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.

Courtney E. Yaeger, Dario L. Ringach, and Joshua T. Trachtenberg. Neuromodulatory control of localized dendritic spiking in critical period cortex. *Nature*, 56, 2019. ISSN 0028-0836. doi: 10.1038/s41586-019-0963-3. URL http://www.nature.com/articles/s41586-019-0963-3.

Alan Yuille and Daniel Kersten. Vision as Bayesian inference: analysis by synthesis? *Trends in Cognitive Sciences*, 10(7):301–308, 2006. ISSN 13646613. doi: 10.1016/j.tics.2006.05.002.

Chapter 6: Measuring and regularizing networks in function space

Foreword

Deep learning provides a catalog of concepts for neuroscience. This chapter is about expanding that catalog. It is written for a deep learning audience, not neuroscience *per se*, and was published in a peer-reviewed deep learning conference[8]. The motivating ideas are common to both fields. What are the available learning algorithms for neural networks?

In deep learning, the algorithm of reference is gradient descent. However, one can make better choices than this. As I review in this chapter, most of deep learning now uses different algorithms in the same family (like Adam, RMSprop, or natural gradients).

The insight of these methods is that all parameters are not equal. Some have a bigger effect on the output, and thus on the *function* encoded by the network. Yet gradient descent treats all parameters equally, in a sense, by measuring distances in parameter space. One can do better by using the space of functions (i.e. behavior). This chapter claims that this is a more natural way of thinking about learning.

Plasticity in the brain might be interpreted in this way, as well. Synapses that are very influential upon behavior (like those on a motor unit in the spinal cord) ought to change more slowly than other, less efficacious synapses. A solution to continual learning (not 'catastrophically forgetting' previous things) can also be derived with this function-space approach. As neuroscience discovers the brain's learning algorithms, this concept may help to interpret what is found.

[8] Benjamin, Ari S., David Rolnick, and Konrad Kording. "Measuring and regularizing networks in function space." in *International Conference on Learning Representations,* 2019

Abstract

To optimize a neural network one often thinks of optimizing its parameters, but it is ultimately a matter of optimizing the function that maps inputs to outputs. Since a change in the parameters might serve as a poor proxy for the change in the function, it is of some concern that primacy is given to parameters but that the correspondence has not been tested. Here, we show that it is simple and computationally feasible to calculate distances between functions in a L^2 Hilbert space. We examine how typical networks behave in this space, and compare how parameter ℓ^2 distances compare to function L^2 distances between various points of an optimization trajectory. We find that the two distances are nontrivially related. In particular, the L^2/ℓ^2 ratio decreases throughout optimization, reaching a steady value around when test error plateaus. We then investigate how the L^2 distance could be applied directly to optimization. We first propose that in multitask learning, one can avoid catastrophic forgetting by directly limiting how much the input/output function changes between tasks. Secondly, we propose a new learning rule that constrains the distance a network can travel through L^2-space in any one update. This allows new examples to be learned in a way that minimally interferes with what has previously been learned. These applications demonstrate how one can measure and regularize function distances directly, without relying on parameters or local approximations like loss curvature.

Introduction

A neural network's parameters collectively encode a function that maps inputs to outputs. The goal of learning is to converge upon a good input/output function. In analysis, then, a researcher should ideally consider how a network's input/output function changes relative to the space of possible functions. However, since this space is not often considered tractable, most techniques and analyses consider the parameters of neural networks. Most regularization techniques, for example, act directly on the parameters (e.g. weight decay, or the implicit constraints stochastic gradient descent (SGD) places upon movement). These techniques are valuable to the extent that parameter space can be taken as a proxy for function space. Since the two might not always be easily related, and since we ultimately care most about the input/output function, it is important to develop metrics that are directly applicable in function space.

In this work we show that it is relatively straightforward to measure the distance between two networks in function space, at least if one chooses the right space. Here we examine L^2-space, which is a Hilbert space. Distance in L^2 space is simply the expected ℓ_2 distance between the outputs of two functions when given the same inputs. This computation relies only on function inference.

Using this idea of function space, we first focus on characterizing how networks move in function space during optimization with SGD. Do random initializations track similar trajectories? What happens in the overfitting regime? We are particularly interested in the relationship between trajectories in function space and parameter space. If the two are tightly coupled, then parameter change can be taken as a proxy for function change. This common assumption (e.g. Lipschitz bounds) might not always be the case.

Next, we demonstrate two possibilities as to how a function space metric could assist optimization. In the first setting we consider multitask learning, and the phenomenon of catastrophic forgetting that makes it difficult. Many well-known methods prevent forgetting by regularizing how much the parameters are allowed to shift due to retraining (usually scaled by a precision matrix calculated on previous tasks). We show that one can instead directly regularize changes in the input/output function of early tasks. Though this requires a ”working memory” of earlier examples, this scheme turns out to be quite data-efficient (and more so than actually retraining on examples from old tasks).

In the second setting we propose a learning rule for supervised learning that constrains how much a network's function can change any one update. This rule, which we call *Hilbert-constrained gradient descent* (HCGD), penalizes each step of SGD to reduce the magnitude of the resulting step in L^2-space. This learning rule thus changes the course of learning to track a shorter path in function space. If SGD generalizes in part because large changes to the function are prohibited, then this rule will have advantages over SGD. Interestingly, HCGD is conceptually related to the natural gradient. As we derive, the natural gradient can be viewed as resulting from constrains changes in a function space measured by the Kullbeck-Leibler divergence.

Results

Examining networks in function space

We propose to examine the trajectories of networks in the space of functions defined by the inner product $\langle f, g \rangle = \int_x f(x) g(x) d\,\mu(x)$, which yields the following norm:

$$\|f\|^2 = \int_{\mathbb{X}} |f|^2 d\mu.$$

Here μ is a measure and corresponds to the probability density of the input distribution X. Note that this norm is over an empirical distribution of data and not over the uniform distribution of all possible inputs. The $|\cdot|^2$ operator refers to the 2-norm and can apply to vector-valued functions. While we refer to this space as a Hilbert space, we make no use of an inner product and can also speak of this as any normed vector space, e.g. a Banach space. This norm leads to a notion of distance between two functions f and g given by

$$\|f - g\|^2 = \int_{\mathbb{X}} |f - g|^2 d\mu.$$

Since μ is a density, $^{\text{R}}{}_{\text{X}}\, d\mu = 1$, and we can write

$$\|f - g\|^2 = \mathbb{E}_{\mathbb{X}}[|f(x) - g(x)|^2].$$

The expectation can be approximated as an empirical expectation over a batch of examples drawn from the input distribution:

$$\|f - g\|^2 \approx \frac{1}{N} \sum_{i=0}^{N} |f(x_i) - g(x_i)|^2$$
.

The quality of the empirical distance, of course, will depend on the shape and variance of the distribution of data as well as the form of f and g. In section 2.3, we empirically investigate the quality of this estimator for reasonably sample sizes N.

Divergence of networks in L^2 space during training

We wish to compare at high level how networks move through parameter and function space. Our first approach is to compare a low-dimensional embedding of the trajectories through these spaces. In Figure 1, we take a convolutional neural network and train three random initializations on a 5000-image subset of CIFAR-10. By saving the parameters of the network at each epoch as well as the output on a single large validation batch, we can later compute the ℓ^2 parameter distance and the L^2 function distance between the snapshots of network at each epoch. The resulting distance matrix is then visualized as a two-dimensional embedding.

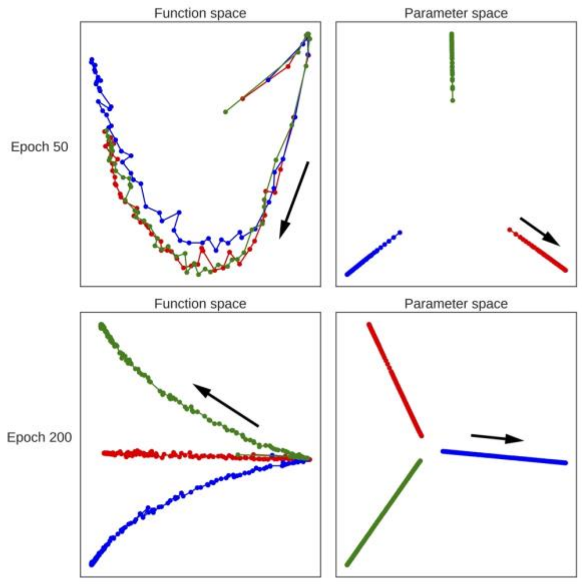


Figure 1: Visualization of the trajectories of three random initializations of a network through function space, left, and parameter space, right. The network is a convolutional network trained on a 5,000 image subset of CIFAR-10. At each epoch, we compute the L^2 and ℓ^2distances between all previous epochs, forming two distance matrices, and then recompute the 2D embedding from these matrices using multidimensional scaling. Each point on the plots represents the network at a new epoch of training. The black arrows represent the direction of movement.

In parameter space, the networks are initialized at very different points and proceed to diverge yet further from these points. Despite this divergence, each trajectory yields a network that has learned the training data perfectly and generalizes with $\sim 50\%$ accuracy to a test set. This illustrates the wide range of parameter settings that can be used to represent a given neural network function. The behavior of the same initializations in function space is quite different. First, note that all three initializations begin at approximately the same point in function space. This is an intriguing property of random initializations that, rather than encoding entirely random functions, random sets of parameters lead on average to the same function (for related work, see e.g. Giryes et al. (2016)). The initializations then largely follow an identical path for the initial stage of learning. Different initializations thus learn in similar manners, even if the distance between their parameters diverges. During late-stage optimization, random initializations turn from a shared trajectory and begin to diverge in L^2 space. These differences underlie the general principle that L^2 distances behave differently than ℓ^2 distances, and that functional regularization could assist training and reduce overfitting.

Comparing function distance with parameter distance

How well do parameter distances reflect function distances? The answer to this question is relevant for any method that directly considers the behavior of parameters. Certain theoretical analyses, furthermore, desire bounds on function distances but instead find bounds on parameter distances and relate the two with a Lipschitz constant (e.g. Hardt et al. (2015)). Thus, for theoretical analyses and optimization methods alike, it is important to empirically evaluate how well parameter distances correspond to function distances in typical situations.

We can compare these two situations by plotting a change in parameters $||\Delta\theta||$ against the corresponding change in the function $|| f_\theta - f_{\theta+\Delta\theta}||$. In Figure 2 we display this relation for several relevant distance during the optimization of a CNN on CIFAR-10. There are three scales: the distance between individual updates, the distance between epochs, and the distance from initialization.

Note, first, that networks continue to move in function space as well as in parameter space after test error converges, which is around epoch 60. (The test error can be seen in Appendix A, along with identical plots colored by test error instead of epoch.) Their movement relative to initialization slows, but there is still large movement relative to previous iterations and previous epochs.

What changes strikingly throughout optimization is the relationship between parameter and function space. There is a qualitative difference in the ratio of parameter distances to function distances that visible at all three distance scales. Early epochs generally see larger changes in L^2 space for a given change in parameters. Intriguingly, the ratio of the two distances appears to converge to a single value at late optimization, after test error saturates. This is not because the network ceases to move, as noted above. Rather, the loss landscape shifts such that this ratio become constant.

It is also clear from these plots that there is not a consistent positive correlation between the parameter and function distances between any two points on the optimization trajectory. For example, the parameter distance between successive epochs is negatively correlated with the L^2 distance for most of optimization (Fig. 2b). The distance from initialization shows a clean and positive relationship, but the relationship changes during optimization. Between successive batches, L^2 distance correlates with parameter distance at late epochs, but less so early in optimization when learning is quickest. Thus, at different stages of optimization, the L^2/ℓ^2 ratio is often quite different.

The usage of Batch Normalization (BN) and weight decay in this analysis somewhat affects the trends in the L^2/ℓ^2 ratio. In Appendix A we reproduce these plots for networks trained without BN and without weight decay. The overall message that the L^2/ℓ^2 ratio changes during optimization is unchanged. However, these methods both change the scale of updates, and appear to do so differently throughout optimization, and thus some trends are different. In Appendix B, we also isolate the effect of training data, by reproducing these plots for a CNN trained on MNIST and find similar trends. Overall, the correspondence between parameter and function distances depends strongly on the context.

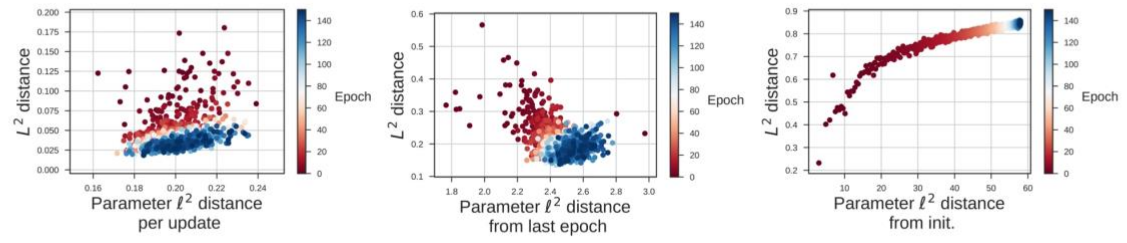


Figure 2: Parameter distances is sometimes, but not always, representative of function distances. Here we compare the two at three scales during the optimization of a CNN on CIFAR-10. Left: Distances between the individual SGD updates. Middle: Distances between each epoch. Right: Distances from initialization. On all three plots, note the changing relationship between function and parameter distances throughout optimization. The network is the same as in Figure 1: a CNN with four convolutional layers with batch normalization, followed by two fully-connected layers, trained with SGD with learning rate = 0.1, momentum = 0.9, and weight decay = 1e-4. Note that the L^2 distance is computed from the output after the softmax layer, meaning possible values range from 0 to 1.

Convergence of the empirical estimator

It might be worried that since function space is of infinite dimension, one would require prohibitively many examples to estimate a function distance. However, we find that one can compute a distance between two functions with a relatively small amount of examples. Figure 3 shows how the estimated L^2 distance converges with an increasing number examples. In general, we find that only a few hundred examples are necessary to converge to an estimation within a few percent.

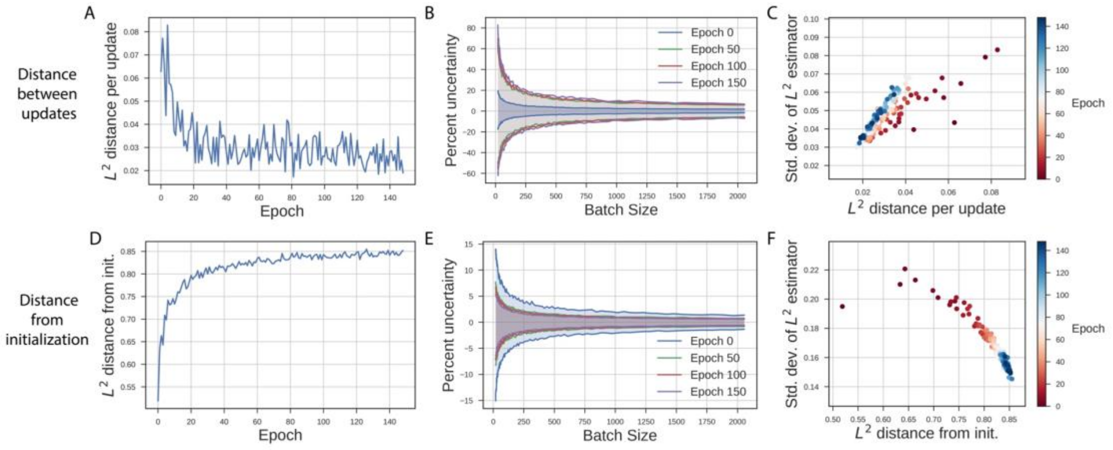


Figure 3: The variance of the L^2 estimator is small enough that it can be reasonably estimated from a few hundred examples. In panels A and D, we reproduced L^2 distances seen in the panels of Fig. 2. As we increase the number of validation examples these distances are computed over, the estimations become more accurate. Panels B and E show the 95% confidence bounds for the estimation; on 95% of batches, the value will lie between these bounds. These bounds can be obtained from the standard deviation of the L^2 distance on single examples. In panel C we show that the standard deviation scales linearly with the L^2 distance when measured between updates, meaning that a fixed batch size will often give similar percentage errors. This is not true for the distance from initialization, in panel F; early optimization has higher variance relative to magnitude, meaning that more examples are needed for the same uncertainty. In the Appendix, we also display the convergence of the L^2 distance estimator between epochs.

Applications

Combatting catastrophic forgetting in an online learning task

If, after having been trained on a task, a neural network is retrained on a new task, it often forgets the first task. This phenomenon is termed 'catastrophic forgetting'. It is the central difficulty of multitask training as well as applications requiring that learning be done online (especially in non-IID situations). Essentially, new information must be encoded in the network, but the the information pertinent to the previous task must not be overwritten.

Most efforts to combat catastrophic forgetting rely on restricting how much parameters can change between tasks. Elastic Weight Consolidation (EWC; Kirkpatrick et al. (2017)), for example, adds a penalty to the loss on a new task B that is the distance from the weights after learning on an earlier . task A, multiplied by the diagonal of the Fisher information matrix F (calculated on task A):

$$L_{EWC}(\theta) = L_B(\theta) + \frac{\lambda}{2}\sum_i F_i(\theta_i - \theta_{i,A})^2$$

This idea is closely related to well-studied approaches to Bayesian online learning, if *F* is interpreted as a precision matrix (Honkela & Valpola (2003), Opper & Winther (1998)). Other similar approaches include that of Ritter et al. (2018), who use a more accurate approximation of the Fisher, and Synaptic Intelligence (SI; Zenke et al. (2017)), which discounts parameter change via a diagonal matrix in which each entry reflects the sum contribution of that parameter to the loss. Each of these method discourages catastrophic forgetting by restricting movement in parameter space between tasks, scaled by a (perhaps diagonal) precision matrix calculated on previous tasks.

Using a function space metric, it is not hard to ensure that the network's output function on previous tasks does not change during learning. In this case, the loss for a new task B is modified to be:

$$L(\theta) = L_B(\theta) + \frac{\lambda}{2}\|f_{\theta_A} - f_{\theta_B}\|$$

The regularization term is the L2 distance between the current function fθB and the function after training on task A, fθA. Since our function space metric is defined over a domain of examples, we will store a small set of previously seen examples in a working memory, as well as the output on those examples. This memory set will be used to calculate the L2 distance between the current iteration and the snapshot after training. This is a simple scheme, but novel, and we are not aware of direct precedence in the literature.

A working memory approach is employed in related work (Lopez-Paz et al. (2017); Rebuffi et al. (2017)). Note, however, that storing old examples violates the rules of strict online learning. Nevertheless, for large networks it will be more memory-efficient. EWC, for example, requires storing a snapshot of each parameter at the end of the previous task, as well as a diagonal precision matrix with as many entries as parameters. For the 2 hidden layer network with 400 nodes each that was used in the MNIST task in Kirkpatrick et al. (2017), this is 1,148,820 new parameters, or as many pixels as 1,465 MNIST images. When each layer has as many as 2,000 nodes, as in Fig. 3B of Kirkpatrick et al. (2017), the extra stored parameters are comparable to 15,489 MNIST images. The working memory approach that is required to regularize function change from old tasks is thus comparable or cheaper in memory.

Empirical results

We compared the performance of our approach at the benchmark task of permuted MNIST. This task requires a single MLP to learn to classify a sequence of MNIST tasks in which the pixels have been randomly permuted differently on each task. We trained an MLP with 2 hidden layers, 400 nodes each, for 10 epochs on each of 8 such permuted datasets. In Figure 4, we display how the test accuracy on the first of 8 tasks degrades with subsequent learning.

To build the working memory, we keep 1024 examples from previous tasks, making sure that the number of examples from each task is equal. We also remember the predictions on those examples at the end of training on their originating tasks. To calculate the L^2 distance, we simply re-infer on the examples in working memory, and regularize the distance from the current outputs to the remembered outputs. We chose $\lambda = 1.3$ as the regularizing hyperparameter from a logarithmic grid search.

In Figure 4, we compare this method to four comparison methods. The ”ADAM” method is ADAM with a learning rate of 0.001, which nearly forgets the first task completely at the end of the 8 tasks. The ”ADAM+retrain” method is augmented with a working memory of 1024 examples that are stored from previous tasks. Every n iterations (we found n = 10 to be best), a step is taken to decrease the loss on the memory cache. This method serves as a control for the working memory concept. We also include EWC and SI as comparisons, using the hyperparameters used in their publications $(\lambda = 500, \epsilon = c = 0.1$). Overall, we found that regularizing the L2 distance on a working memory cache was more successful than simply retraining on the same cache. It also outperformed EWC, but not SI. Note that these methods store diagonal matrices and the old parameters, and in this circumstance these were larger in memory than the memory cache.

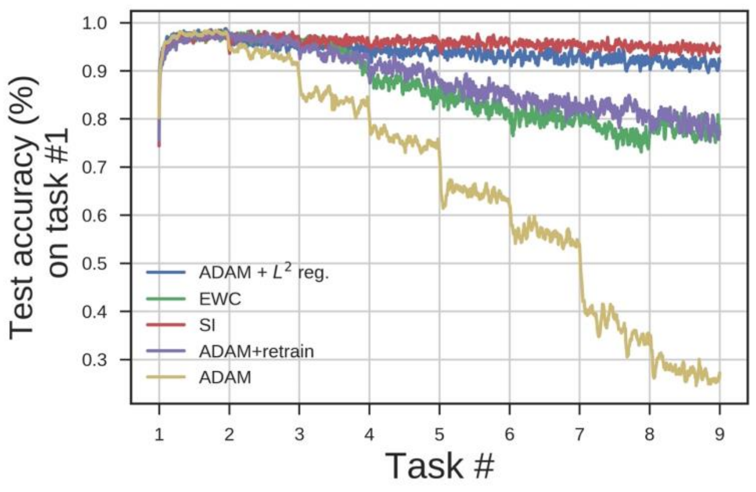


Figure 4: Regularizing the L^2 distance from old tasks (calculated over a working memory cache of size 1024) can successfully prevent catastrophic forgetting. Here we display the test performance on the first task as 7 subsequent tasks are learned. Our method outperforms simply retraining on the same cache (ADAM+retrain), which potentially overfits to the cache. Also displayed are ADAM without modifications, EWC, and SI.

Constraining changes in L² during learning

In this section we propose that the L^2 distance can be used for regularization in a single supervised task. In the space of parameters, SGD is a strongly local update rule and large jumps are generally prohibited. SGD is thus more likely to find solutions that are close to the initialization, and furthermore to trace a path of limited length. This discourages the sampling a large volume of parameter space during optimization. If the mapping between parameter and function space is not already very tight, and locality is important for generalization, then additionally constricting changes in function space should help.

On the basis of this logic, we propose a learning rule that directly constrains the path length of the optimization trajectory L^2 space. If a network would have been trained to adjust the parameters θ to minimize some cost C_0, we will instead minimize at each step t a new cost given by:

$$C = C_0 + \lambda || f_{\theta_t} - f_{\theta_t + \Delta\theta} ||$$

Like all regularization terms, this can also be viewed as a Langrangian that satisfies a constraint. Here, this constraint ensures that the change in L^2-space does not exceed some constant value. To evaluate Equation 1, we can approximate the norm with an empirical expectation over X:

$$C = C_0 + \lambda \Big(\frac{1}{N} \sum_{i=0}^{N} |f_{\theta_t}(x_i) - f_{\theta_t + \Delta\theta}(x_i)|^2 \Big)^{1/2}.$$

This cost function imposes a penalty upon the difference between the output of the current network at time t and the proposed network at $t + 1$. The data x_i may derive from some validation batch but must pull from the same distribution X. It would also be possible to use unlabeled data.

We can write an update rule to minimize Equation 1 that is a modification of gradient descent. We call the rule Hilbert-constrained gradient descent (HCGD). It minimizes C in Equation 1 via an inner loop of gradient descent. To optimize C via gradient descent, we first replace C_0 with its first order approximation $J^T\Delta\theta$, where J is the Jacobian. Thus we seek to converge to a $\Delta\theta^0$ at each update step, where

$$\Delta\theta' = \underset{\Delta\theta}{\operatorname{argmin}} \left(J^T \Delta\theta + \frac{\lambda}{N} \sum_{i=0}^{N} |f_{\theta_t}(x_i) - f_{\theta_t + \Delta\theta}(x_i)|^2 \right) \quad (2)$$

Minimization of the proper $\Delta\theta$ can be performed in an inner loop by a first order method. We first propose some $\Delta\theta_0 = -\epsilon J = -\epsilon \nabla_\theta C_0$ (for learning rate) and then iteratively correct this proposal by gradient descent towards $\Delta\theta^0$. If only one correction is performed, we simply add the derivative of the Hilbert-constraining term after $\Delta\theta_0$ has been proposed. We found empirically that a single correction was often sufficient. In Appendix C, we demonstrate that this algorithm does actually decrease the distance traveled in function space, as expected. This algorithm is shown in Algorithm 1.

Algorithm 1: Hilbert-constrained gradient descent. Implements Equation 2.

```
Require: $\epsilon$                                                  ▷ Overall learning rate
Require: $\eta$                                    ▷ Learning rate for corrective step
 1: procedure
 2:     $\theta \leftarrow \theta_0$                                   ▷ Initialize parameters
 3:     while $\theta_t$ not converged do
 4:         draw $X \sim \mathbb{P}_x$                                ▷ Draw training batch
 5:         $J \leftarrow \nabla_\theta C_0(X)$                        ▷ Calculate gradients
 6:         $\Delta\theta_0 \leftarrow -\epsilon J$                    ▷ Proposed update

 7:         draw$X_V \sim \mathbb{P}_x$                             ▷ Draw validation batch
 8:         $g_{L^2} \leftarrow \nabla_{\Delta\theta}\big(\frac{\lambda^2}{N}\sum_{x_i \in X_V}^{N} |f_{\theta_t}(x_i) - f_{\theta_t+\Delta\theta}(x_i)|^2\big)^{1/2}$   ▷ Calculate correction
 9:         $\Delta\theta' \leftarrow \Delta\theta_0 - \eta(g_{L^2})$
10:         $\theta_t \leftarrow \theta_{t-1} + \Delta\theta'$
11:     return $\theta_t$
```

Note that the "proposed update" is presented as an SGD step, but could be a step of another optimizer (e.g. ADAM). In the Appendix, we display an extended version of this algorithm. This version allows for multiple corrective iterations in each step. It also allows for a form of momentum. In standard momentum for SGD, one follows a "velocity" term *v* which is adjusted at each step with the rule $v \leftarrow \beta v + \epsilon J$ (e.g. see Sutskever et al. (2013)). For HCGD, we also keep a velocity term but update it with the final Hilbert-constrained update $\Delta\theta$ rather than *J*. The velocity is used to propose the initial $\Delta\theta_0$ in the next update step. We found that this modification of momentum both quickened optimization and lowered generalization error.

Relation to the natural gradient

The natural gradient turns out to carry a similar interpretation as HCGD, in that the natural gradient also regularizes the change in functions' output distributions. Specifically, the natural gradient can be derived from a penalty upon the change in a network's output distribution as measured by the Kullbeck-Leibler divergence (rather than the L^2 distance).

To show this, we start with a similar goal of function regularization and will come upon the natural gradient. Let us seek to regularize the change in a network's output distribution P_θ throughout optimization of the parameters θ, choosing the Kullbeck-Leibler (KL) divergence as a measure of similarity between any two distributions. To ensure the output distribution changes little throughout optimization, we define a new cost function

$$C = C_0 + \lambda D_{KL}(P_{\theta_{t+1}}||P_{\theta_t}) \qquad (3)$$

where C_0 is the original cost function and λ is a hyperparameter that controls the importance of this regularization term. Optimization would be performed with respect to the proposed update θ_{t+1}.

Evaluating the KL divergence directly is problematic because it is infeasible to define the output density P_θ everywhere. One can obtain a more calculable form by expanding $D_{KL}(P_{\theta_{t+1}} k P_{\theta_t})$ around θ_t to second order with respect to θ. The Hessian of the KL divergence is the Fisher information metric F. With $\Delta\theta \equiv (\theta_{t+1} - \theta_t)$, we can rewrite our regularized cost function as

$$C \approx C_0 + \frac{\lambda}{2}\Delta\theta^T F \Delta\theta \qquad (4)$$

To optimize C via gradient descent we first replace C_0 with its first order approximation.

$$C \approx J^T \Delta\theta + \frac{\lambda}{2}\Delta\theta^T F \Delta\theta \qquad (5)$$

At each evaluation, J is evaluated before any step is made, and we seek the value of $\Delta\theta$ that minimizes Equation 5. By setting the derivative with respect to $\Delta\theta$ to be zero, we can see that this value is

$$\Delta\theta = \frac{1}{\lambda} F^{-1} J \qquad (6)$$

When λ = 1 this update is equal to the natural gradient. Thus, the natural gradient emerges as the optimal update when one regularizes the change in the output distribution during learning.

In Appendix E, we show how one can approximate the natural gradient with an inner first-order optimization loop, like in HCGD. We note that HCGD is computationally cheaper than the exact natural gradient. It does not require any matrix inversions, nor the calculation of separate per-example gradients. When the validation batch X_V is drawn anew for each of n corrective iterations (step 8 in Algorithm 1), HCGD requires an additional two forward passes and one backwards pass for each correction, for a total of 2 + 3n passes each outer step.

The natural gradient in the literature

In addition to being seen as a regularizer of functional change, it in an interesting aside to note that variants of the natural gradient have appeared with many justifications. These include data efficiency, minimizing a regret bound during learning, speeding optimization, and the benefits of whitened gradients.

Amari originally developed the natural gradient in the light of information geometry and efficiency (Amari et al. (1996); Amari (1998)). If some directions in parameter space are more informative of the network's outputs than others, then updates should be scaled by each dimension's informativeness. Equivalently, if not all examples carry equal information about a distribution, then the update step should be modified to make use of highly informative examples. That is, we wish to find a Fisher-efficient algorithm (see Amari et al. (2000)). The natural gradient uses the Fisher information matrix to scale the update by parameters' informativeness.

There is also a connection between the natural gradient (and thus HCGD) and techniques that normalize and whiten gradients. The term $F^{-1}J$, after all, simply ensures that steps are made in a parameter space that is whitened by the covariance of the gradients. Whitening the gradients thus has the effect that SGD becomes more similar to the natural gradient. It appears that many approaches to normalize and whiten activations or gradients have been forwarded in the literature (Raiko et al. (2012);Simard et al. (1998); Schraudolph & Sejnowski (1996); Crammer et al. (2009); Wang et al. (2013); LeCun et al. (1991); Schraudolph (1998); Salimans & Kingma (2016)). A similar effect is able to be learned with Batch Normalization, as well (Ioffe & Szegedy (2015)). By normalizing and whitening the gradients, or by proxy, the activations, these various methods ensure that parameter space is a better proxy for function space.

Empirical comparison of HCGD

We compared HCGD and SGD on feedforward and recurrent architectures. If it is important that SGD limits changes in function space, and parameter and function space are loosely coupled, then HCGD should improve upon SGD. In all tests, we used a tuned learning rate for SGD, and then used the same learning rate for HCGD. We use values of $\lambda = 0.5$ and $\eta = 0.02$, generally about 10 times less than the principal learning rate. (For the $n = 1$ version, λ can be folded into the inner learning rate η. Values were chosen so that $\lambda\eta = 0.01$.) We chose the batch size for the "validation" batch to be 256. While the examples in each "validation" batch were different than the training batch, they were also drawn from the train set. All models were implemented in PyTorch (Paszke et al. (2017)).

We tested HCGD as applied to the CIFAR-10 image classification problem. For reproducibility, we trained a Squeezenet v1.1, a convolutional neural network model with batch normalization optimized for parameter efficiency (Iandola et al. (2016)). Overall HCGD does not outperform SGD in the final learning stage when trained with the same learning rate as SGD (initial$\epsilon = 0.1$), though it does perform better in the early stage while the learning rate is high (Figure 5). When we increase the initial learning rate to$\epsilon = 0.3$ (red trace), the training accuracy decreases but the test accuracy is still marginally higher than SGD. Given the difference in relative performance between the high and low learning rate stages, it is possible that HCGD requires a different learning rate schedule to achieve the same level of gradient noise. HCGD thus decreases the test error at a given learning rate, but needs to be trained at a higher learning rate to achieve the same level of gradient noise.

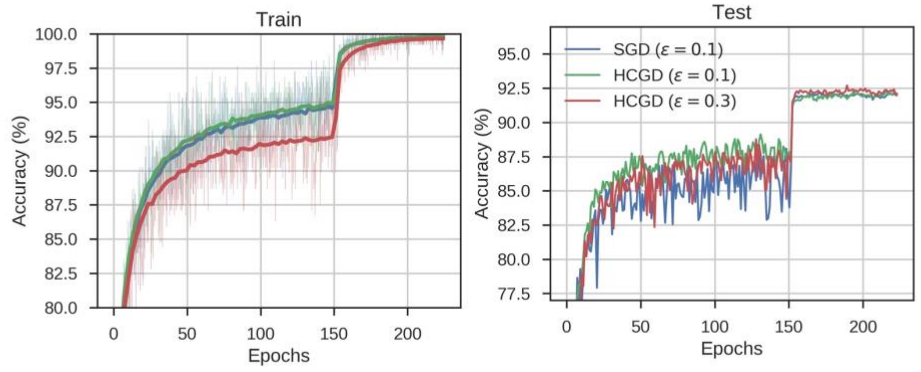


Figure 5: Results of a Squeezenet v1.1 trained on CIFAR10. The learning rate is decreased by a factor of 10 at epoch 150. For the train error we overlay the running average of each trace for clarity.

We next tested the performance of HCGD on a recurrent task. We trained an LSTM on the sequential MNIST task, in which pixels are input one at a time. The order of the pixels was permuted to further complicate the task. We found that HCGD outperformed SGD (Figure 6. We used 1 correction step, as before, but found that using more correction steps yielded even better performance. However, HCGD underperformed ADAM. While not the ideal optimizer for this task, the fact that SGD can be improved indicates that SGD does not move as locally in function space as it should. Parameter space thus a poor proxy for function space in recurrent networks.

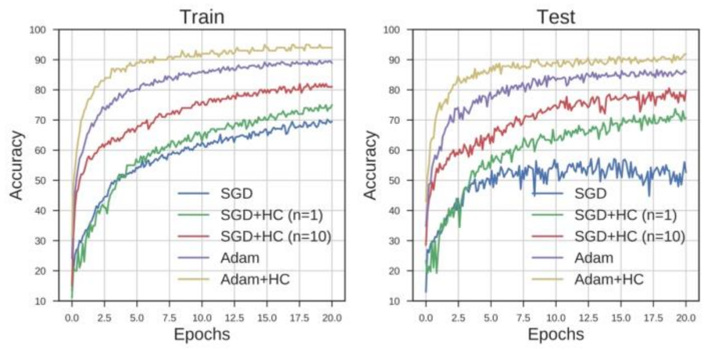


Figure 6: Results of a singlelayer LSTM with 128 hidden units trained on the sequential MNIST task with permuted pixels. Shown are the traces for SGD and Adam (both with learning rate 0.01). We then take variants of the HCGD algorithm in which the first proposed step is taken to be an SGD step (SGD+HC) or an Adam step (Adam+HC). For SGD+HC we also show the effect of introducing more iterations n in the SGD+HC step.

HCGD first proposes an update by SGD, and then corrects it, but the first update step can also be other optimizers. Since Adam worked well for the sequential MNIST task, we tested if Adam could also be improved by taking a step to penalize the change in function space. We found that this is indeed the case, and show the results as well in Figure 6. To differentiate the SGD- and Adam-based methods, we refer to in the figure as SGD+HC and Adam+HC. This combination of Adam and L^2 functional regularization could help to achieve state-of-the-art performance on recurrent tasks.

Discussion

Neural networks encode functions, and it is important that analyses discuss the empirical relationship between function space and the more direct parameter space. Here, we argued that the L^2 Hilbert space defined over an input distribution is a tractable and useful space for analysis. We found that networks traverse this function space qualitatively differently than they do parameter space. Depending on the situation, a distance of parameters cannot be taken to represent a proportional distance between functions.

We proposed two possibilities for how the L^2 distance could be used directly in applications. The first addresses multitask learning. By remembering enough examples in a working memory to accurately estimate an L^2 distance, we can ensure that the function (as defined on old tasks) does not change as a new task is learned. This regularization term is agnostic to the architecture or parameterization of the network. We found that this scheme outperforms simply retraining on the same number of stored examples. For large networks with millions of parameters, this approach may be more appealing than comparable methods like EWC and SI, which require storing large diagonal matrices.

We also proposed a learning rule that reduces movement in function space during single-task optimization. Hilbert-constrained gradient descent (HCGD) constrains the change in L^2 space between successive updates. This approach limits the movement of the encoded function in a similar way as gradient descent limits movement of the parameters. It also carries a similar intuition as the forgetting application: to learn from current examples only in ways that will not affect what has already been learned from other examples. HCGD can increase test performance at image classification in recurrent situations, indicating both that the locality of function movement is important to SGD and that it can be improved upon. However, HCGD did not always improve results, indicating either that SGD is stable in those regimes or that other principles are more important to generalization. This is by no means the only possibility for using an L^2 norm to improve optimization. It may be possible, for example, to use the norm to regularize the confidence of the output function (e.g. Pereyra et al. (2017)). We are particularly interested in exploring if more implicit, architectural methods, like normalization layers, could be designed with the L^2 norm in mind.

It interesting to ask if there is support in neuroscience for learning rules that diminish the size of changes when that change would have a large effect on other tasks. One otherwise perplexing finding is that behavioral learning rates in motor tasks are dependent on the direction of an error but independent of the magnitude of that error (Fine & Thoroughman, 2006). This result is not expected by most models of gradient descent, but would be expected if the size of the change in the output distribution (i.e. behavior) were regulated to be constant. Regularization upon behavioral change (rather than synaptic change) would predict that neurons central to many actions, like neurons in motor pools of the spinal cord, would learn very slowly after early development, despite the fact that their gradient to the error on any one task (if indeed it is calculated) is likely to be quite large. Given our general resistance to overfitting during learning, and the great variety of roles of neurons, it is likely that some type of regularization of behavioral and perceptual change is at play.

References

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Shun-ichi Amari, Andrzej Cichocki, and Howard Hua Yang. A new learning algorithm for blind signal separation. In *Advances in neural information processing systems*, pp. 757–763, 1996.

Shun-Ichi Amari, Hyeyoung Park, and Kenji Fukumizu. Adaptive method of realizing natural gradient learning for multilayer perceptrons. *Neural Computation*, 12(6):1399–1409, 2000.

Koby Crammer, Alex Kulesza, and Mark Dredze. Adaptive regularization of weight vectors. In *Advances in neural information processing systems*, pp. 414–422, 2009.

Michael S Fine and Kurt A Thoroughman. Motor adaptation to single force pulses: sensitive to direction but insensitive to within-movement pulse placement and magnitude. *Journal of neurophysiology*, 96(2):710–720, 2006.

Raja Giryes, Guillermo Sapiro, and Alexander M Bronstein. Deep neural networks with random Gaussian weights: a universal classification strategy? *IEEE Trans. Signal Processing*, 64(13): 3444–3457, 2016.

Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

Antti Honkela and Harri Valpola. On-line variational bayesian learning. In *4th International Symposium on Independent Component Analysis and Blind Signal Separation*, pp. 803–808, 2003.

Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters andi 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, pp. 201611835, 2017.

Yann LeCun, Ido Kanter, and Sara A Solla. Eigenvalues of covariance matrices: Application to neural-network learning. *Physical Review Letters*, 66(18):2396, 1991.

David Lopez-Paz et al. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pp. 6467–6476, 2017.

James Martens. New perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pp. 2408–2417, 2015.

Manfred Opper and Ole Winther. A bayesian approach to on-line learning. *On-line Learning in Neural Networks, ed. D. Saad*, pp. 363–378, 1998.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

Tapani Raiko, Harri Valpola, and Yann LeCun. Deep learning made easier by linear transformations in perceptrons. In *Artificial Intelligence and Statistics*, pp. 924–932, 2012.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proc. CVPR*, 2017.

Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. *arXiv preprint arXiv:1805.07810*, 2018.

Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 901–909, 2016.

Nicol Schraudolph. Accelerated gradient descent by factor-centering decomposition. 1998.

Nicol N Schraudolph and Terrence J Sejnowski. Tempering backpropagation networks: Not all weights are created equal. In *Advances in neural information processing systems*, pp. 563–569, 1996.

Patrice Simard, Yann LeCun, John Denker, and Bernard Victorri. Transformation invariance in pattern recognitiontangent distance and tangent propagation. *Neural networks: tricks of the trade*, pp. 549–550, 1998.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147, 2013.

Chong Wang, Xi Chen, Alexander J Smola, and Eric P Xing. Variance reduction for stochastic gradient optimization. In *Advances in Neural Information Processing Systems*, pp. 181–189, 2013.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. *arXiv preprint arXiv:1703.04200*, 2017.

Appendix

A CIFAR-10 L^2 AND ℓ^2 COMPARISON

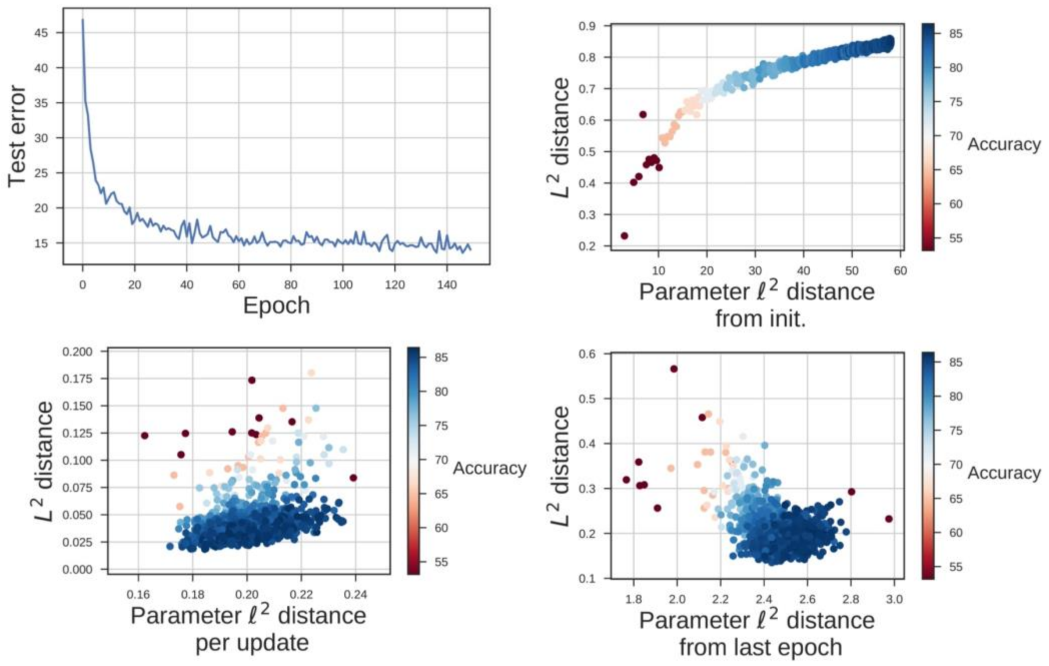


Figure A.1: This figure reproduces Figure 2, but includes the test error. The color scale is now also the test accuracy, rather than epoch number. Note that those epochs with qualitatively different L^2/ℓ^2 ratios than the late optimization correspond to the epochs where test error is changing fastest.

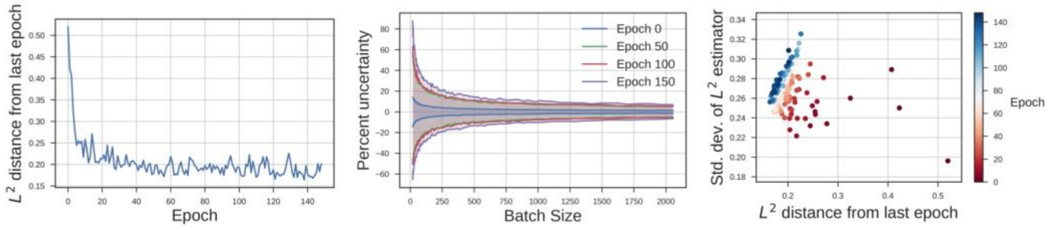


Figure A.2: This figure completes Figure 3 to include the standard deviation of the estimator for the distance between epochs. The scale of the standard deviation is similar to that of the L^2 estimator between batches, requiring near 1,000 examples for accuracies within a few percent.

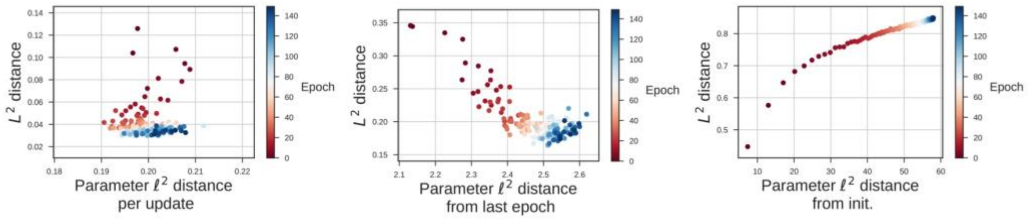


Figure A.3: Same as Figure 2 (L^2/ℓ^2 ratio for three distance scales) but with all points within an epoch averaged. This makes the overall trends more apparent.

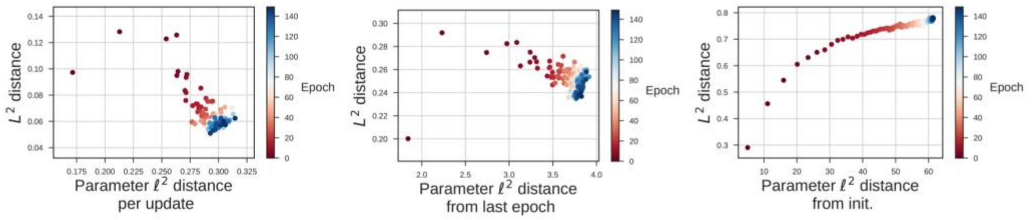


Figure A.4: Same as above, but for a network trained without Batch Normalization (BN). The change is most apparent in the x-axis scale of the left and middle plots. Without BN, larger parameter changes yield the same magnitude of L^2 changes, both between updates and between epochs. Furthermore, the L^2/ℓ^2 ratio for the distance between updates (leftmost plot) changes less between epochs when BN is used. This appears largely a consequence of BN keeping the typical update size fixed at a more standard magnitude (and yet achieving a similar functional change.

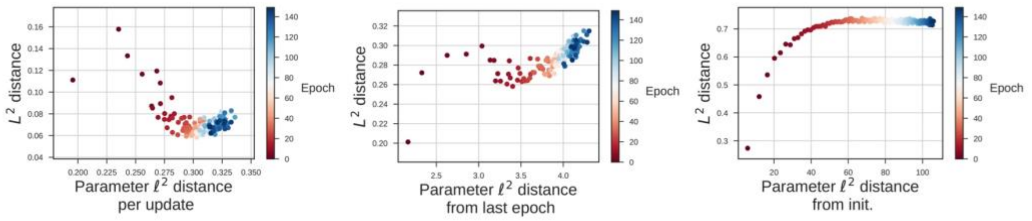


Figure A.5: Same as above, but for a network trained without Batch Normalization and also without weight decay. Weight decay has a strong effect. The main effect is that decreases the ℓ^2 distance traveled at all three scales (from last update, last epoch, and initialization), especially at late optimization. This explains the left column, and some of the middle and right columns. (It is helpful to look at the "white point" on the color scale, which indicates the point halfway through training. Note that parameter distances continue to change after the white point when WD is not used). An additional and counterintuitive property is that the L^2 distance from the last epoch increases in scale during optimization when WD is not used, but decreases if it is. These comparisons show that

WD has a strong effect on the L^2/ℓ^2 ratio, but that this ratio still changes considerable throughout training. This is in line with this paper's motivation to consider L^2 distances directly.

B COMPARING FUNCTION AND PARAMETER SPACES DURING MNIST OPTIMIZATION

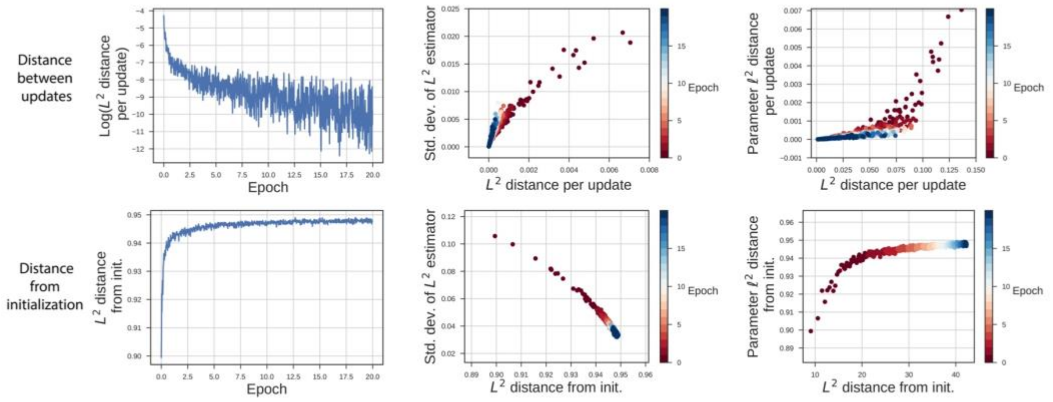


Figure B.6: Here we reproduce the results of Figure 2 and Figure 3 for the MNIST task, again using a CNN with batch normalization trained with SGD with momentum. It can be seen first that the majority of function space movement occurs very early in optimization, mostly within the first epoch. The standard deviation of the L^2 estimator, which sets the number of examples needed to accurately estimate a consistent value, is somewhat higher than for CIFAR-10. Finally, at right, it can be seen that the relationship between parameter distance traveled and function distance is similar to that of a CNN on CIFAR-10, include the qualitative change after test error converges (which here is around epoch 1).

C HCGD DECREASES THE DISTANCE TRAVELED IN L^2 SPACE

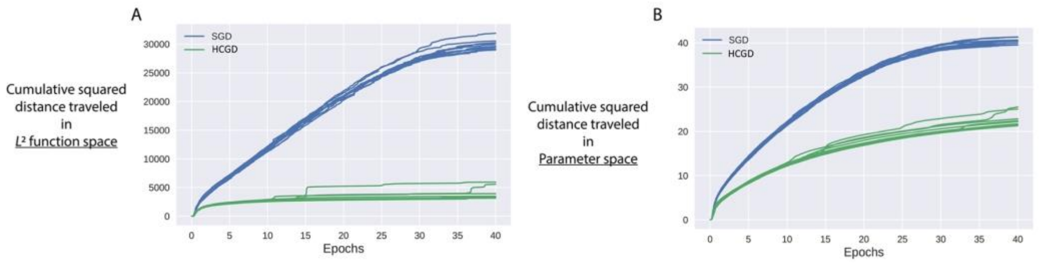


Figure C.7: The HCGD algorithm is designed to reduce motion through L2-space. To confirm this, here we plot the cumulative squared distance traveled during optimization for a simple MLP trained on MNIST. This is calculated by the simple cumulative sum of the squared distances between consecutive updates. (The squared distance is nice because Brownian motion will present as a linear increase in its cumulative sum). It can be seen that SGD continues to drift in L2-space during the overfitting regime (around epoch 15, which is when test error saturates), while HCGD plateaus. This indicates that the function has converged to a single location; it ceases to change. With SGD, on the other hand, the network continues to cahnge even long after test error saturates. It is interesting to note that HCGD allows the parameters to continue to drift even though the function has generally converged.

D DETAILED HCGD ALGORITHM

This version of the algorithm includes momentum. It also allows for multiple corrections.

Algorithm 2: Hilbert-constrained gradient descent. Implements Equation 7.

```
Require: $n \geq 1$                                   ▷ Number of corrective steps. May be 1.
Require: $\epsilon$                                   ▷ Overall learning rate
Require: $\eta$                                       ▷ Learning rate for corrective step
Require: $\beta$                                      ▷ Momentum
 1: procedure
 2:     $\theta \leftarrow \theta_0$                  ▷ Initialize parameters
 3:     $v \leftarrow 0$                              ▷ Initialize momentum buffer
 4:     while $\theta_t$ not converged do
 5:         reset dropout mask, if using
 6:         draw $X \sim \mathbb{P}_x$                ▷ Draw training batch
 7:         $J \leftarrow \nabla_\theta C_0(X)$       ▷ Calculate gradients
 8:         $v \leftarrow \beta v + \epsilon J$
 9:         $\Delta\theta_0 \leftarrow -v$            ▷ Proposed update

10:         draw$X_V \sim \mathbb{P}_x$               ▷ Draw validation batch
11:         $g_{L^2} \leftarrow \nabla_{\Delta\theta}\big(\frac{\lambda^2}{N}\sum_{i=0}^{N}|f_{\theta_t}(x_i) - f_{\theta_t+\Delta\theta}(x_i)|^2\big)^{1/2}$
12:         $\Delta\theta_1 \leftarrow \Delta\theta_0 - \eta(g_{L^2})$     ▷ First correction
13:         $v \leftarrow v + \eta(g_{L^2})$          ▷ Update buffer
14:         for $1 < j < n$ do                        ▷ Optional additional corrections
15:             $g_{L^2} \leftarrow J + \nabla_{\Delta\theta}\big(\frac{\lambda^2}{N}\sum_{i=0}^{N}|f_{\theta_t}(x_i) -$
16:                 $f_{\theta_t+\Delta\theta_{j-1}}(x_i)|^2\big)^{1/2}$
17:             $\Delta\theta_j \leftarrow \Delta\theta_{j-1} - \eta(g_{L^2})$
18:             $v \leftarrow v + \eta(g_{L^2})$
19:         $\theta_t \leftarrow \theta_{t-1} + \Delta\theta$
20:     return $\theta_t$
```

E NATURAL GRADIENT BY GRADIENT DESCENT

In order to better compare the natural gradient to the Hilbert-constrained gradient, we propose a natural gradient algorithm of a similar style.

Previous work on the natural gradient has aimed to approximate F^{-1} as best and as cheaply as possible. This is equivalent to minimizing Equation 2 (i.e. $J\Delta\theta + \frac{\lambda}{2}\Delta\theta^T F\Delta\theta$) with a single iteration of a second-order optimizer. For very large neural networks, however, it is much cheaper to calculate matrix-vector products than to approximately invert a large matrix. It is possible that the natural gradient may be more accessible via an inner gradient descent, which would be performed during each update step as an inner loop.

We describe this idea at high level in Algorithm 2. After an update step is proposed by a standard optimizer, the algorithm iteratively corrects this update step towards the natural gradient. To start with a good initial proposed update, it is better to use a fast diagonal approximation of the natural gradient (such as Adagrad or RMSprop) as the main optimizer. Each additional correction requires just one matrix-vector product after the gradients are calculated. Depending on the quality of the proposed update, the number of iterations required is likely to be small, and even a small number of iterations will improve the update.

Algorithm 3: Natural gradient by gradient descent. This algorithm can be paired with any optimizer to increase its similarity to the natural gradient.

```
Require: $n$                                        ▷ Number of corrective steps. May be 1.
Require: $\eta$                                     ▷ Learning rate for corrective step
1: procedure
2:     $\theta \leftarrow \theta_0$                                    ▷ Initialize parameters
3:     while $\theta_t$ not converged do
4:         $\Delta\theta_0 \leftarrow \mathrm{RMSprop}(\theta_t)$          ▷ Use any optimizer to get proposed update
5:         for $i < n$ do                                ▷ Corrective loop
6:             $\Delta\theta_{i+1} = \Delta\theta_i - \eta(J + \lambda F \Delta\theta_i)$    ▷ Step towards $\frac{1}{\lambda}F^{-1}J$
7:         $\theta \leftarrow \theta + \Delta\theta$
8:     return $\theta_t$
```

Since the Fisher matrix F can be calculated from the covariance of gradients, it never needs to be fully stored. Instead, for an array of gradients G of size (# parameters, # examples), we can write

$$F\Delta\theta = (GG^T)\Delta\theta = G(G^T\Delta\theta) \quad (7)$$

The choice of G is an important one. It cannot be a vector of aggregated gradients (i.e. J), as that would destroy covariance structure and would result in a rank-1 Fisher matrix. Thus, we must calculate the gradients on a per-example basis. To compute G efficiently it is required that a deep learning framework implement forward-mode differentiation, which is currently not supported in popular frameworks.

If we choose G to be the array of per-example gradients on the minibatch, F is known as the 'empirical Fisher'. As explained in Martens (2014) and in Pascanu & Bengio (2013), the proper method is to calculate G from the predictive (output) distribution of the network, $P_\theta(y|x)$. This can be done as in Martens & Grosse (2015) by sampling randomly from the output distribution and re-running backpropagation on these fictitious targets, using (by necessity) the activations from the minibatch. Alternatively, as done in Pascanu & Bengio (2013), one may also use unlabeled or validation data to calculate G on each batch.

Chapter 7: Efficient neural codes naturally emerge through gradient descent learning

Foreword

This chapter focus on how sensory representations are shaped by learning. In a collaboration with Alan Stocker's lab[9], we showed how many aspects of visual perception can be explained with ideas from the study of learning in artificial neural networks. In this chapter, machine learning plays **Role 4**, a model of the brain.

This chapter links two frameworks in two fields: efficient coding (in sensory neuroscience) and the implicit biases of gradient descent (in deep learning theory). Both create similar sensory representations. This link is surprising, and can explain why artificial neural networks trained on natural image tasks appear in many ways similar to neural responses in the ventral stream (reviewed in Chapter 4, Role 4).

We do not treat this this result as evidence that the brain performs gradient descent, although is suggestive. Many algorithms might lead to similar outcomes. However, it is a powerful proof of principle of an alternative type of explanation for neuroscience. Sensory representations might be explained by appealing not just to evolution, but to the process of sensory learning in infancy. Whatever algorithm the brain uses to generalize from limited experience, it will shape sensory representations in the adult.

Deep learning theory has a great deal to offer neuroscience beyond this finding. Future work may help to describe why adults and infants alike learn what they do.

[9] Ari Benjamin, Ling-Qi Zhang, Cheng Qiu, Alan Stocker, Konrad Kording. "Efficient neural codes naturally emerge through gradient descent learning" ." bioRxiv (2022) *https://doi.org/10.1101/2022.05.11.491548*

Abstract

Animal sensory systems are more sensitive to common features in the environment than uncommon features. For example, small deviations from the more frequently encountered horizontal orientations can be more easily detected than small deviations from the less frequent diagonal ones. Here we find that artificial neural networks trained to recognize objects also have patterns of sensitivity that match the statistics of features in images. To interpret these findings, we show mathematically that learning with gradient descent in deep neural networks preferentially creates representations that are more sensitive to common features, a hallmark of efficient coding. This result suggests that efficient coding naturally emerges from gradient-like learning on natural stimuli.

Introduction

Careful psychophysical studies of perception has revealed that neural representations do not encode all aspects of stimuli with equal sensitivity (Fechner, 1948). The ability to detect a small change in a stimulus, for instance, depends systematically on stimulus value. A classic example of this is the so-called ‘oblique effect’ in which changes in visual orientation are easier to detect near vertical or horizontal than oblique orientations (Appelle, 1972). The fact that these sensitivity patterns are ubiquitous and widely shared between animals motivates us to study the potential underlying reasons why they exist.

The efficient coding hypothesis has become a standard explanation for the emergence of these non-homogeneous sensitivity patterns (Barlow, 1961). It predicts that sensory systems should preferentially encode more common aspects of the world at the expense of less common aspects, as this is the most efficient way (in the information-theoretical sense) to make use of limited coding resources. Indeed, perceptual sensitivity typically reflects the statistics of the visual environment (Coppola et al, 1998; Ganguli and Simoncelli, 2010; Girshick et al, 2011; Wei and Stocker, 2015, 2017). While much is known about efficient neural codes and their link to the stimulus statistics and perceptual behavior, the mechanisms that give rise to such codes remain unknown.

Brains are not born with fully developed sensory representations. Many developmental studies of perception in infants and young children have shown that visual sensitivities improve with age and visual experience even until adolescence (Armstrong et al, 2009; Braddick and Atkinson, 2011; Mayer and Dobson, 1982; Teller and Movshon, 1986). The maturing optics of the developing eye and retina explain some of this improvement, but much of the improvement depends on visual experience and is due to neural changes downstream (Banks and Crowell, 1993; Maurer et al, 1999; Movshon and Kiorpes, 1993; Gold et al, 1999; Schoups et al, 2001). This suggests that perceptual sensitivity depends on the neural representation of sensory information and how these representations change with experience during development.

We hypothesize that general, task-oriented learning is the mechanism that gives rise to efficient sensory representations in the brain. Any gradual learning process can only learn so much at a time. This means that an effective learning algorithm should prioritize learning more important aspects before less important ones. Conveniently, features that are more common are also easier to learn from a limited exposure to the world. This suggests that effective learning rules for neural networks might naturally produce better representations for common features, hence providing a mechanism for the emergence of efficient codes. This phenomenon is equivalent to the notion that learning provides a second, implicit constraint on neural coding in addition to the explicit constraint imposed by the limited neural resources. Our hypothesis directly predicts that we should find forms of efficient coding not just in biological neural networks but also in other learning systems. Especially, we expect artificial neural networks trained to perform visual recognition tasks to exhibit efficient neural representations similar to those found in the visual cortex of biological brains despite their many differences in their local structural properties (e.g. noise) and connectivity.

A study of the consequences of effective yet gradual learning on sensory representations must begin from a specific learning rule. One canonical learning rule is gradient descent, which proposes that neural updates are as small as possible to elicit a given improvement in behavior. Though the brain may use more complicated learning rules, gradient descent is arguably the simplest rule for general learning and thus a baseline for theorizing about learning in the brain. If gradient descent produces efficient codes, this would provide a strong proof of principle that efficient codes can emerge from general-purpose learning algorithms.

To show that efficient coding emerges from gradient descent requires a formal understanding of how learning with gradient descent biases what is represented about the stimulus. This parallels an active effort in the study of deep learning. It is now recognized that what neural network learn about their inputs is constrained not just by their connectivity but also implicitly by their learning algorithm (Zhang et al, 2021). Many potential implicit constraints have been propose due to their importance in explaining why large neural networks work well on unseen data (i.e. generalize) (Jacot et al, 2018; Neyshabur et al, 2017; Smith and Le, 2017; Tishby and Zaslavsky, 2015). One prominent theory is gradient descent in a multilayer network produces nontrivial learning dynamics that supplies key biases on what is learned first (Arora et al, 2019; Gidel et al, 2019; Gunasekar et al, 2018; Razin and Cohen, 2020; Saxe et al, 2013). This raises the possibility that such ideas could also demonstrate whether gradient descent learning is biased towards efficient codes.

In this paper, we demonstrate that learning with gradient descent biases feature learning towards common input features, thus reproducing the relationship between stimulus statistics and perceptual sensitivity (Fig. 1). This effect occurs in otherwise unconstrained and noiseless networks as well as for multiple learning objectives (i.e. not limited to information maximization). We examine two model systems. First, we show that deep artificial networks trained on natural image classification show similar patterns of sensitivity as humans, and that this is a partly a consequence of image statistics (also see Benjamin et al (2019); Henderson and Serences (2021)) but is also partially due to factors inherent in network architecture. We then leverage results from the study of linear networks to mathematically describe how gradient descent naturally causes learned representations to reflect the input statistics. To demonstrate that this framework can be applied to explain development, we also show that changes in sensitivity resembling changes in visual acuity in human children can be reproduced in a simple model trained with gradient descent on natural images. Our results show how learning provides a natural mechanism for the emergence of a non-uniform sensory sensitivity that matches input statistics.

Results

Humans and animals show sensitivity that depends on the orientation of stimuli. In humans, the sensitivity of internal representations can be inferred from psychophysical data on discrimination thresholds (Ganguli and Simoncelli, 2010) or the empirical distribution of tuning curves in V1 (Schoups et al, 2001; Stringer et al, 2021) (Fig. 2a). In many animals, internal representations are most sensitive at near vertical and horizontal orientations (Appelle, 1972). This raises the question if neural networks trained on natural stimuli are similarly sensitive in a non-uniform way.

To ask if neural networks would show a similar phenomenon, we first obtained a set of relevant networks and measured their response to artificial stimuli. We specifically investigated deep neural networks trained on the ImageNet task (Deng et al, 2009) as such networks show a number of other similarities to human ventral streamvisual processing (Gü¸clü and van Gerven, 2015; Khaligh-Razavi and Kriegeskorte, 2014; Yamins et al, 2014). We analyzed a range of architectures, including two large convolutional neural networks (CNNs), VGG16 and Resnet18, and Vision Transformers which operate largely without convolution (Dosovitskiy et al, 2020; He et al, 2016; Simonyan and Zisserman, 2014). Then, to measure sensitivity, we measured the squared magnitude of the change in network activations given a change in the angle of oriented Gabor stimuli (Fig 2b; see Methods). For all three networks, we found that the internal representations were most sensitive to changes near cardinal orientations (Fig. 2c). The effect was more pronounced deeper in each network. The coarse pattern of sensitivity of ImageNet-trained deep networks to orientation is thus similar to that of animals and animals.

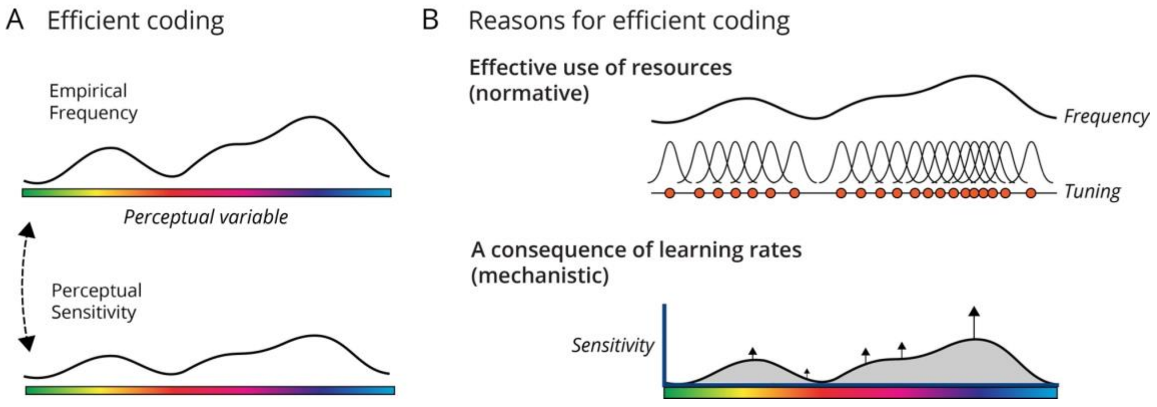

A Efficient coding
Empirical Frequency
Perceptual variable
Perceptual Sensitivity
B Reasons for efficient coding
Effective use of resources (normative)
Frequency
Tuning
A consequence of learning rates (mechanistic)
Sensitivity

Fig. 1 Reasons for efficient coding. A) One consequence of efficient coding is that perceptual sensitivity reflects the empirical frequency of perceptual variables. B) Efficient coding can be justified normatively as the most effective way to allocate finite neural resources to encode a stimulus ensemble. In this work we describe a mechanism for efficient coding due to learning components of the inputs at different rates dependent on their frequency.

We next investigated whether this pattern was due to factors inherent in the network or due to the statistics of the inputs on which it was trained. Before training, randomly initialized networks largely do not show this pattern, nor do networks in which all weights after training are randomly shuffled (SI Fig. 1a). After training networks on a version of ImageNet in which all images are rotated by 45º, the networks lose sensitivity to cardinals and gain sensitivity to oblique angles (SI Fig. 1b). This finding recapitulates our preliminary findings and concurrent work of colleagues, and points to an origin in image statistics (Benjamin et al, 2019; Henderson and Serences, 2021). However, we also found that networks trained on rotated images do partially retain sensitivity to cardinal orientations; they do not simply rotate their sensitivity by 45º (SI Fig. 1b). This indicates that increased sensitivity to cardinals is partially due to factors inherent in the convolutional architecture. Indeed, we found that the use of spatial pooling with overlapping receptive fields (such as in AlexNet; Krizhevsky et al (2012)) involves oversampling a rectangular grid and that this produces a significant cardinal sensitivity (SI Fig 1c). The pattern of orientation sensitivity is thus both a product of the input statistics and inherent factors like architecture.

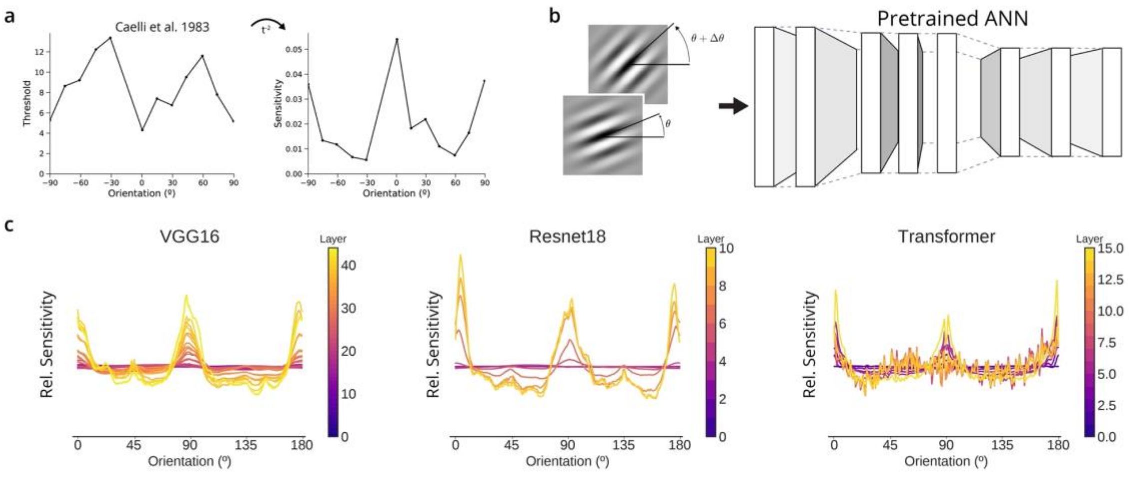


Fig. 2 Artificial neural networks trained to classify naturalistic images show similar patterns of sensitivity as humans. a) Discrimination thresholds for orientation vary systematically in humans. The sensitivity of the underlying internal representations, as the Fisher Information, can be inferred as the inverse square of the threshold (Ganguli and Simoncelli, 2010; Wei and Stocker, 2017). b) We measured the sensitivity of each layer in an artificial network as the change in layer's response due to a given change in orientation, i.e. the squared norm of the gradient with respect to orientation. c) Relative (normalized) sensitivity to orientation for three networks trained on ImageNet, plotted for various layers in each network.

To separate effects related to architecture and learning, we next examined the sensitivity of artificial neural networks to changes in hue, as this is unlikely to be affected by rectangular convolutional processing. We found that hue sensitivity after training was related to the empirical frequency of hues in ImageNet (Fig. 3c) measured in HSV color space. The location of the peaks of network sensitivity roughly matched the patterns of human sensitivity to changes in hues in the HSV color space (Fig. 3b). To test if this pattern is causally related to the input statistics, we trained a Resnet18 network on a version of ImageNet in which the hue of all pixels was shifted by 90º and observed a corresponding shift in the hue sensitivity (Fig. 3d). This suggests that in general the frequency of low-level visual features determines the sensitivity of artificial neural networks trained on object classification.

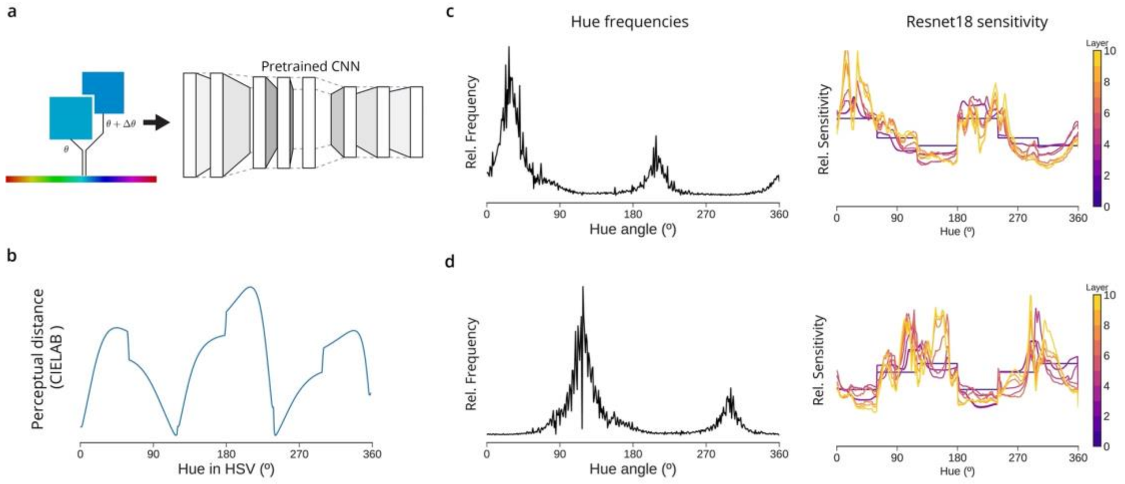


Fig. 3 The sensitivity of ANNs to hue also matches image statistics. a) The color of a uniform image is varied; in HSV color space, saturation and value are held fixed and the hue is varied. Results are averaged over possible saturations and values. b) In humans, the sensitivity to the H axis can be inferred by the perceptual distance between uniformly spaced H values (calculated using the approximately perceptually uniform color space CIELAB) at S=V=1. c) The sensitivity to hue in each layer in a trained ResNet18 tracks the empirical frequency of hues in the ImageNet dataset. d) Training ResNet18 on a version of ImageNet in which hues are rotated results in a corresponding shift in hue sensitivity.

Sensitivity may track input statistics in artificial networks for a similar reason as in humans. In psychophysics, one leading explanation proposes that there is some constraint that limits the amount of information a neural population can contain about its inputs. Due to this constraint, an optimal code will allocate more resources (and be more sensitive) to inputs that occur frequently (Ganguli and Simoncelli, 2010; Wei and Stocker, 2016). However, the networks above are overparameterized, in the sense that internal layers contain a greater number of nodes than there are input pixels, and furthermore contain no source of information-limiting noise during evaluation. Thus, the frequency/sensitivity correspondence in artificial networks likely does not arise from an optimal encoding of the inputs despite inherent and unresolvable architectural constraints.

One alternative possibility to an explicit constraint like noise is that incremental learning via gradient descent naturally leads to a frequency/sensitivity correspondence. This hypothesis relates to the idea from connectionist models of development that the most general aspects of a problem are often learned first (Munakata and McClelland, 2003). To investigate this possibility, we analyzed a category of artificial neural networks amenable to mathematical study: deep linear networks. Deep linear networks contain no nonlinearities and are equivalent to sequential matrix multiplication. Despite their simplicity, deep linear networks show many of the same learning phenomena as nonlinear networks (Saxe et al, 2013; Arora et al, 2019) and humans (Lee et al, 2014; Saxe et al, 2019). Moreover, this simplified setting allows us to separate the effects of gradient descent from those of network nonlinearity.

How can we characterize learning in this linear setting? At a high level, the network becomes responsive to features earlier when those features are more common (Fig. 4). When learning ends due to finite training time, finite data, or saturating performance, there is a residual higher sensitivity for common features. We will expand on this phenomenon below. Overall, the link between learning rate and input frequency, combined with finite training time, is an additional inductive bias beyond what features are useful for the task and means that trained networks will tend to be more sensitive to frequent features.

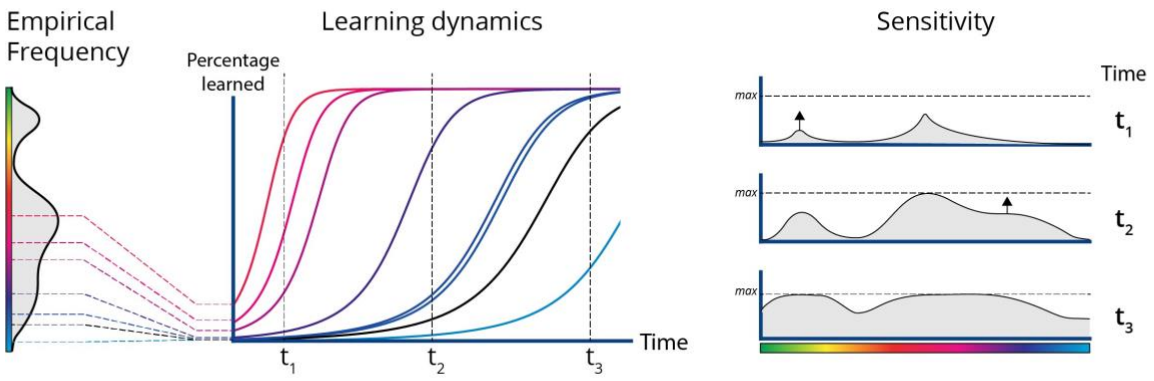


Fig. 4 Schematic of how the learning dynamics of linear networks causes a correspondence between network sensitivity and input statistics. The learning problem is broken into components, each of which learns at a specific rate. The frequency or variance of a feature of the input data (e.g. the color red) in part determines the learning rate of the components that encode it. This means that the network becomes sensitive to frequent features first. Training may end before all features are fully encoded.

To more concretely demonstrate this phenomenon we will focus on the task of reconstructing natural images with a linear network (Fig. 5). This network can be as shallow as a single layer, in which case the reconstructed images are given by the matrix multiplication $X^{\wedge} = WX$. Importantly, this problem can be solved exactly with the solution that the weight matrix W is the identity matrix I. This is an unconstrained problem; if there is any non-uniformity in the sensitivity of the output $X^{\wedge}$ to changes in X it must be due to the implicit constraints posed during learning. Analyzing the output sensitivity in this simple model will help to better understand the implicit preferences of learning with gradient descent.

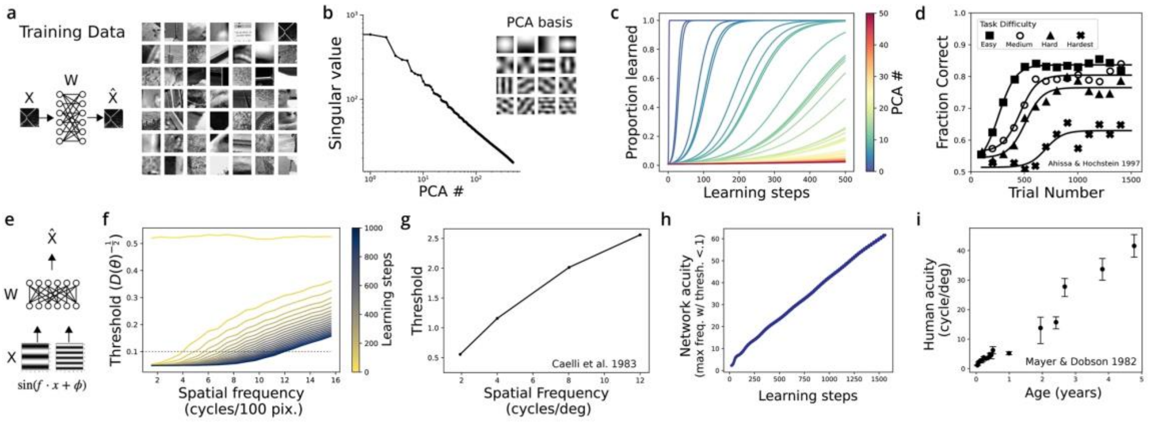


Fig. 5 The effect of input statistics on network sensitivity can be understood with linear network models. Despite their simplicity these show human-like learning phenomena. a) We trained linear networks to reconstruct black and white patches of natural images. b) The statistics (here, variance) of each PC is given by its singular value, which for natural images shows a characteristic power law decay. c) When learning with gradient descent, the weight matrix learns each PC separately and in order of their variance. The sharpness of the sigmoidal learning curve is controlled by the network depth (SI Fig. 2) d) Human perceptual learning curves are also sigmoidal, and increasing task difficulty delays learning dynamics. Data replotted from Ahissar and Hochstein (1997); subjects trained to detect the orientation of a line, and the difficulty of the task was controlled by a masking stimulus. e-i) Sensitivity to spatial frequency. f) Every 50 learning steps we plotted the inverse square root of the sensitivity to spatial frequency, which is a proxy for detection thresholds. At each step note the linear increase above an elbow. g) Human data on spatial frequency thresholds replotted from Caelli et al (1983). h) An artificial spatial 'acuity' grows nearly linearly with training; 'acuity' is defined as the maximum spatial frequency for which the artificial threshold is below a value of 0.1. i) In infants and children, the spatial acuity – the highest spatial frequency observable for high-contrast gratings – increases linearly with age. Replotted from Mayer and Dobson (1982).

In our demonstrative task we will examine the sensitivity of the output $X^{\wedge}$ to the magnitude of each principal component that makes up an image (as provided by PCA on the inputs, Fig. 5b). The mathematical analysis for this feature is much simpler than, say, for orientation. In this case also we have some expectation as to what pattern of sensitivity the efficient coding framework predicts because the principal components (PCs) are ordered by their variance. An efficient code in the presence of independent internal noise should be more sensitive to earlier PCs. Indeed, earlier PCs are composed of lower spatial frequencies, and humans are better at detecting changes in lower spatial frequencies (Fig. 5e). If gradient descent provides a similar effect, we should find that the output $X^{\wedge}$ becomes sensitive to lower PCs first.

In our linear model it is possible to describe analytically how the sensitivity changes due to gradient descent. This is done by examining how the weights change. We first decompose the weights W via singular value decomposition (SVD), $W = USV^T = \sum_i \sigma_i u_i v_i^T$, as a product of unit-length singular vectors (u,v) and their corresponding singular values σ_i. The evolution of these components under gradient descent is known as long as certain basic conditions are met, such as a very small weight initialization (Arora et al, 2019; Saxe et al, 2013). One key previous finding is that the singular vectors v of the weight matrix rotate to align with the PCs of the input (see Theorem 2 in the Appendix) (Arora et al, 2019). Due to this alignment, the sensitivity of the output $X^{\wedge}$ to the ith PC is controlled by the size of the corresponding singular value in the weights, σ_i. This is more formally derived in Methods. If σ_i remains near its initialization close to zero, then the projection of data upon the ith PC will be filtered out and the output will not be sensitive to the corresponding PC. The sensitivity of $X^{\wedge}$ to each PC and how it changes with learning can be understood entirely by the growth of the singular values of W.

Thus far we have linked sensitivity to the weights, but it remains to input statistics to how the weights change. The input statistics affect the growth of the singular values of the weight matrix. Each σ_i grows at a different rate. For this objective of reconstruction, the growth rate of σ_i is proportional to the standard deviation of the corresponding ith PC in the data (see Methods). These decay as a power law for natural images and are shown in the spectrum in Fig. 5b. As a result, the network output will become sensitive first to the first (largest-variance) PCs and later to the later PCs. This is verified empirically in Fig. 5c. Only at infinite training times does the weight matrix encode all PCs equally and recover the exact solution $W = I$. We thus see that, at any finite learning time, the output of the linear network will be more sensitive to the earlier PCs. Note that this non-uniformity in sensitivity emerges despite the lack of any constraints on W other than learning.

Having introduced this model to explain our findings in artificial networks, we next wondered how it would compare to human behavioral data. We first examined the sensitivity of the linear model to the spatial frequency of a sinusoidal grating (Fig. 5e). In adult humans, the detection threshold to changes in frequency increases linearly with frequency (Fig. 5g) (Caelli et al, 1983). To compare to human data, we can plot the "detection threshold" of an artificial network as the inverse squared sensitivity of the network output to frequency. This is proportional to the error rate of an optimal read-out of frequency given injected Gaussian noise (Rao, 1945). At several snapshots during training, we observed that the spatial frequency threshold increased linearly with frequency above a certain cutoff frequency, below which the threshold saturated at a low value (Fig. 5f). Even a single matrix trained to reconstruct images reproduces human-like sensitivity to spatial frequency when trained with gradient descent.

If the human perceptual system is also implicitly constrained by learning dynamics, this would be apparent in psychophysical studies of young children. Indeed, the highest observable frequency of a sinusoidal grating continues to improve with age even up to adolescence (Fig. 5i) (Leat et al, 2009; Mayer and Dobson, 1982). This is experience-dependent; when sight is restored in young children by the removal of cataracts their spatial acuity gradually improves (Maurer et al, 1999). These effects can be reproduced in our model of linear image reconstructions. By measuring the network's spatial acuity as a function of training episode as the highest spatial frequency whose simulated detection threshold (inverse squared sensitivity) was below a fixed cutoff, we found we could reproduce a linear increase of spatial acuity with age (Fig. 5h). Learning with gradient descent reproduces not only an efficient encoding of spatial frequency but also the way in which visual acuity increases linearly with age.

The theory of learning in deep networks makes several further predictions for human perception, many of which have been explored previously (Saxe, 2015; Wenliang and Seitz, 2018). A central feature of this framework is a characteristic sigmoidal curve for perceptual learning tasks (Fig. 5c). Such sigmoidal learning curves are observable in humans on perceptual learning tasks that are sufficiently difficult (Fig. 5d) (Ahissar and Hochstein, 1997). This curve is sigmoidal because the rate of improvement depends upon the current level of sensitivity as well as the difference from asymptotic sensitivity (see Methods). This causes sensitivity to rise exponentially at first but eventually converge exponentially towards an asymptote. In human perceptual learning experiments, the learning curve is indeed better described as an exponential than other functional forms such as power laws (Dosher and Lu, 2007). Although gradient descent in linear systems is a simple model, it accurately captures the functional form of how perception improves with experience.

In the analysis above we trained towards the objective of reconstructing inputs, which is an unsupervised objective. However, the mathematical reason why gradient descent learns frequent inputs first also applies to supervised learning. For two features with equal correlation with the output labels but different variance in the inputs, the network will learn to use the higher-variance feature first (see Methods for derivation). Due to this additional bias beyond task usefulness, networks trained on a wide range of objectives will show greater sensitivity for frequent features.

The emergence of efficient coding in supervised tasks can be verified with a simple task in which the frequency and usefulness of input features are varied independently (Fig. 6). We trained a nonlinear 3-layer neural network to decode the orientation of a sinusoidal grating appearing with a set probability distribution. We also applied noise to the output labels to control the information in each stimulus about the labels. As expected, both the input frequency and output noise separately affect the sensitivity of learned representations (Fig. 6b,c). However, this could also be explained by the effect of frequency and noise on task usefulness, defined as the total information in the input dataset about each label. To demonstrate that gradient descent introduces an additional bias beyond task usefulness, we next adjusted the magnitude of the noise such that the total information is uniform across labels. This requires applying a greater level of noise onto the labels that are more common, balancing their effects on information. Even in this case, a higher sensitivity to input orientation emerges for more common orientations (Fig.

6d). This now provides a deeper intuition of our findings earlier that networks trained on object recognition are more sensitive to frequent features. The preference for frequent features is a general feature of learning with gradient descent and is separate from frequency's effect on the information about labels.

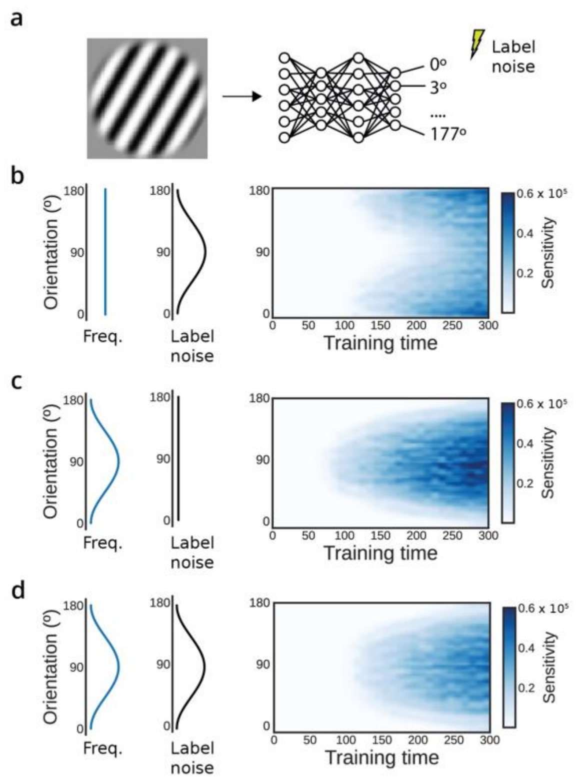


Fig. 6 Dissociating the effect of frequency and information in supervised learning tasks. a) We trained 3-layer nonlinear neural networks to classify the orientation of sinusoidal gratings into 3º bins, varying either input frequency or output noise. b) We controlled the informativeness of input orientations by injecting noise into the labels as a function of orientation. The sensitivity of the first layer to input orientation is shown over learning. With uniform statistics, the more informative features are preferentially learned. c) The effect of varying input frequency without applying label noise. In this case, the more frequent features are preferentially learned. d) We then balanced noise and frequency such that the total information in the input dataset about each output label is uniform (see Methods). Learning with gradient descent still prefers common angles.

Discussion

Here we found that the internal representations of artificial neural networks trained on ImageNet are more sensitive to basic visual features that are more common, which is a hallmark feature of efficient coding. We show that this hallmark naturally emerges from gradient-based learning. Even a minimal model of gradient-based learning – linear image reconstruction – reproduced human patterns of sensitivity to sensory variables and how these change over development. In this minimal model the dynamics of learning can be understood analytically. The correspondence of sensitivity and statistics emerges due to an implicit bias of gradient descent for common, high-variance aspects of the input data.

Our result provides a proof of principle that patterns of perceptual sensitivity in animals could be explained by a similar phenomenon. If plasticity in the brain approximates the gradient of some task, whether by reinforced Hebbian rules or some other algorithm, neural populations will preferentially encode the strongest dimensions in their inputs. As a result, organisms may not need dedicated learning algorithms for efficient codes in addition to algorithms for task-oriented learning. Many such local, unsupervised plasticity rules and neural control mechanisms have been previously proposed as ways the brain might develop efficient codes (Barlow, 1989; Bell and Sejnowski, 1995; Brito and Gerstner, 2016; Hyvärinen and Oja, 1997; Intrator and Cooper, 1992; Karklin and Simoncelli, 2011; Olshausen and Field, 1996; Ruderman and Bialek, 1993; Schwartz and Simoncelli, 2001; Zhou and Yu, 2018). Instead, any general algorithm approximating gradient descent may produce similar codes when learning towards many objectives.

It is important to note that our mathematical analysis of linear networks is highly simplified and may not accurately describe how learning affects sensitivity in general. A number of considerations complicate a generalization to nonlinear artificial neural networks, let alone brains. Nonlinearity makes linear decompositions inaccurate, and as a result we cannot say the precise features that gradient descent prefers to learn before others in nonlinear networks. New techniques from this emerging field may soon allow a more complete characterization of the dynamics of learning (e.g. Goldt et al (2020)). However, despite these caveats, we find that this model is useful to explain why efficient codes emerge in nonlinear artificial networks. It is remarkable that such a simple model of learning also captures qualitative features of human perceptual learning, as well. Learning in linear systems provides a valuable source of intuition for the effects of learning in more complex systems.

While we have shown one mechanism for how learning can induce a statistics/sensitivity correspondence, it is not the only mechanism by which it could do so. Theories of deep learning often distinguish between the "rich" (feature learning) and "lazy" (kernel) regimes possible in network learning (Woodworth et al, 2020). Our models reside in the rich regime, which involves learning new intermediate representations and assumes a small weight initialization. In the alternative, lazy regime, intermediate representations change little over learning and only a readout function is learned (Jacot et al, 2018). Interestingly, networks in the "lazy" (kernel) regime evolve under gradient descent as if they were linear in their parameters (Lee et al, 2019), and furthermore have the inductive bias of successively fitting higher modes of the input/output function as more data is presented (Bordelon et al, 2020; Canatar et al, 2021). The modes are defined differently, however, via the kernel similarity matrix rather than the direct input covariance. Despite the dissimilarities in these mathematical approaches, both center gradient descent and learn important aspects of the problem before others. These similarities suggest that a statistics/sensitivity correspondence could be derived for other network regimes.

Our broadest finding — that task-oriented learning can be a mechanism of producing efficient codes — is relevant to the discussion in psychophysics of the nature of the constraints implied by perceptual sensitivity patterns. It has long been recognized that these patterns imply some limitation upon coding capacity. Here, we make the distinction between implicit limitations due to (a lack of) learning and explicit limitations upon the maximum achievable code quality after learning, such as noise, metabolism, or a limited number of neurons. Although these categories limit perception with different mechanisms, they produce similar patterns of perceptual sensitivity. This distinction is particularly meaningful for explaining perceptual learning. Previously, perceptual improvements during development have been interpreted as a reduction in internal noise accompanied by a continuous maintenance of optimally efficient codes (Dosher and Lu, 1998; Kiorpes and Movshon, 1998). In our framework, learning naturally creates codes that reflect environmental statistics at all stages of learning, and there is no need to invoke a reduction in noise to explain improvements. This is consistent with the view that perceptual learning increases the signal-to-noise ratio through neuronal changes that enhance the signal strength (Gold et al, 1999; Schoups et al, 2001). To be sure, the nervous system is indeed constrained by hard ceilings such as noise and metabolism; learning probably ceases eventually. The implicit constraints due to learning are complementary to these and their relative contribution decreases with age and experience.

Although operating at a much shorter time-scale, sensory adaptation induces similar behavioral changes as perceptual learning, such as improving sensitivity to small stimulus differences at the adapted (i.e. learned) stimulus value. It has been argued that sensory adaptation is a form of efficient coding, optimally re-allocating sensory encoding resources according to recent stimulus history (Fairhall et al, 2001). While the adaptation induced changes in neural encoding characteristics such as reduction in response gain and changes in tuning curves are well characterized (Benucci et al, 2013; Kohn and Movshon, 2004), it is unknown how these local changes in neural representation accuracy depend on the specific details and dynamics of the sensory history. Thus it will be interesting to explore the degree to which sensory adaptation and its dynamics can be explained and predicted by the global objective of a task-dependent learning rule (gradient descent) in a continually updating (i.e., adapting) sensory processing system such as the brain.

The model system of gradient descent in linear systems can make several further predictions if taken as a model of human perceptual learning. In this model, the perceptual learning rate can be quantitatively modeled as a function of input statistics, importance, and current performance. These predictions could be verified in experiments that separably vary label noise and input statistics in supervised perceptual learning problems. Additionally, learning in the rich domain predicts that the learning system should represent the outside world in a low-dimensional way, with additional dimensions being added over time according to their variance and importance. As such, these learning dynamics naturally give rise to low-dimensional neural representations. Such learning dynamics may thus underlie the popular idea in neuroscience of low-dimensional neural manifolds (see Flesch et al (2022)).

A learning framework for perception points to a different sort of normative analysis of why we perceive the way that we do. Optimality can be defined in two ways. It can characterize the maximum achievable code quality, in an information-theoretic sense, given some number of neurons and their biological limitations. Alternatively, one might also describe responses that are optimal given the limited experience by which to learn the statistics of the world. Even ideal observers must learn from limited data, and successful learning from limited data must be constrained (Wolpert and Macready, 1997).

Appropriate learning constraints would be selected for by evolution. Future research may help to unravel these optimal learning algorithms and characterize their sensory consequences.

Methods

Stimuli and calculation of network sensitivity

In all networks, we defined the sensitivity of a particular layer to a sensory variable as the squared magnitude of the gradient. For a layer with N nodes and vector of activations *y*, the sensitivity with respect to a sensory variable θ is:

$$D(y;\ \theta) = \sum_{i}^{N} \frac{\partial y_i}{\partial \theta}^2 .$$

This definition of sensitivity can be related to the Fisher Information about a sensory variable θ in an artificial stimuli set. This is relevant for comparisons to human psychophysical data as the notion of sensitivity inferred from discrimination thresholds is the Fisher Information. In particular, our definition of sensitivity can be interpreted as the Fisher Information of systems with internal Gaussian noise of unit variance, and furthermore for the orientation of stimuli within an artificial stimulus ensemble with one stimulus per value of θ. A derivation of this connection can be found in the Mathematical Appendix.

The sensitivity can be calculated through backpropagation or by the method of finite differences. We created differentiable generators of stimuli in the automatic differentiation framework of Pytorch. This allowed calculating the sensitivity directly via in-built backpropagation methods.

For the figures in the text, we used Gabor stimuli with a spatial frequency of 2 cycles per 100 pixels, a contrast so that pixels span the range of [-1,1] in intensity in units of z-scored ImageNet image intensities, and a Gaussian envelope with σ = 0.5. We marginalized over the phase of the Gabor by averaging the sensitivity calculated with 10 linearly spaced spatial phases tiling the interval $[-\pi,\pi]$. The sinusoidal stimuli input to the linear network varied in spatial frequency, and we similarly averaged sensitivity over spatial phase. Finally, for the hue stimuli, we generated images of a uniform color in HSV color space and converted pixel values to RGB. Results were marginalized over the S and V axes in the range [.5,1] which corresponds to the calculation of hue histograms on ImageNet (see below), which necessarily involves binning S and V.

Deep nonlinear network experiments

To measure the sensitivity of pretrained networks, we first downloaded pretrained ResNet18 and VGG16 (with batch normalization) networks from the Torchvision python package (v0.11) distributed with Pytorch. For the vision transformer, we used a distribution in Python available at https://github.com/lukemelas/PyTorchPretrained-ViT. For each layer in these networks, we calculated sensitivity to orientation and hue were calculated with the stimuli generators described above. The 'layers' are defined differently for each network. For ResNet, layers are what in this architecture are called residual blocks (each of which contain multiple linearnonlinear operations). For VGG, layers are the activations following each linear or pooling layer. Layers within the vision transformer are what are called transformer blocks.

We implemented a number of controls to determine the extent to which the observed patterns of sensitivity related to image statistics. We first ran the sensitivity analysis on untrained networks; we used the Pytorch default initialization. To ensure that the architectures do not show inherent patterns only in a certain regime of weight sizes, we calculated sensitivity on a copy of the networks in which the weights were shuffled. We wanted to preserve weight sizes in a layer-specific manner, and so shuffled the weights only within each tensor.

As further control on the effect of image statistics we retrained certain models on a version of ImageNet in which all images were rotated by 45º, or as well in which the hue of images were rotated by 90º. The transformer model was not retrained due to its expense. Image modifications were performed with Torchvision's in-built rotation and hue adjusting image transformations.

We trained all networks using a standard training procedure: stochastic gradient descent with an initial learning rate of 0.1, decaying by a factor of 10 every 30 epochs, as well as a momentum value of 0.9, and a batch size of 256 images. The networks were trained for 90 epochs. To match the original training setup, we augmented the image dataset with random horizontal reflections and random crops of a size reduction factor varying from 0.08 to 1. Note that the random horizontal reflections change the statistics of orientations so as to be symmetric around the vertical axis. After training the sensitivity was calculated as above.

ImageNet hue statistics

We wrote a custom script to extract the hue histogram of all pixels in all images in the ImageNet training set. We binned hues with a resolution of 1º, and binned hues over the S and V range [.5,1] to focus on strongly colored pixels. The exact range is arbitrary, but importantly matches the range used when calculating network sensitivity.

Linear network experiments

We first constructed a database of 32x32 images of natural scene image portions. These image portions were extracted from ImageNet (Deng et al, 2009), made greyscale, and cropped to size. Our constructed dataset contained over 100,000 examples of image portions. We then performed PCA on this dataset using the PCA method in Scikit-Learn (Pedregosa et al, 2011), and displayed the singular values of the top 1,000 components in Figure 5.

Our task consisted of reconstructing these image portions using a single- or multilayer fully-connected linear neural network. To ensure no architectural bottleneck exists, the internal (hidden) dimension of the multilayer network remained at 32^2, the same as the input and output. The initial parameter values of the networks were scaled down by a factor of 100 from the default Pytorch initialization to ensure rich-regime learning. Networks were trained to minimize the mean-squared error of reconstruction using stochastic gradient descent, a learning rate of 1.0, and a batch size of 16,384, the largest that would fit in memory. The large batch size minimizes effects relating to batch stochasticity.

During learning, we calculated the sensitivity to spatial frequency as well as the projection of the learned weight matrix upon the PCA basis vectors of the inputs. The projection upon each PCA vector is given by $u^T_i Wu_i$, where W is the product matrix corresponding to the linear network and u_i is the ith PCA component. The sensitivity to spatial frequency was calculated by constructing a sinusoidal plane wave test stimulus with parameterized frequency and phase and using Pytorch's automatic differentiation capability to obtain the derivative of network output with respect to frequency. The sensitivity was calculated for 64 equally-spaced phase offsets and the result averaged over phase.

Supervised label noise experiment (Fig. 6)

In this experiment we trained a 3-layer neural network with ReLU nonlinearities to decode the orientation of 64x64 pixel image of a sinusoidal grating. The period of the sinusoid was 12.8 pixels, and in each stimulus the sinusoid carried a random phase offset. The random phase and orientation ensured that no image was repeated. In each image the orientation was sampled in the interval $[0,\pi]$ from a specified probability distribution (either a uniform distribution or $\frac{2-\cos(2x)}{2\pi}$). The objective was the categorization of images into 60 bins of orientations, with success quantified via a cross-entropy loss function.

The addition of noise to the output labels was calibrated such that, on average over a dataset, any orientation θ is as informative as any other despite a potentially nonuniform orientation distribution $p(\theta)$. Since the total information in a dataset about a (potentially noised) label y_θ scales linearly with how often it appears, all else held equal, the variation in per-example information must exactly balance the change in frequency. That is, for any two orientations θ_i and θ_j and their corresponding

(noised) labels $y_{\theta i}$ and $y_{\theta j}$, it must be that $p(\theta_i)I[y_{\theta i} \mid \theta_i] = p(\theta_j)I[y_{\theta j} \mid \theta_j]$. Here $I[\cdot]$ represents the information gained about a label having observed an input, i.e. the change in entropy over y_θ from the uniform distribution. This proportionality is satisfied if $I[y_{\theta_j} \mid \theta_j] = \frac{a}{p(\theta)}$ for some constant a.

Our approach thus requires applying label noise of a known entropy that varies with orientation. Because we optimize a cross-entropy objective, rather than e.g. a mean-squared-error objective, there are no interactions between neighboring bins.

We applied noise by treating the nonzero element of the each label vector, which are indicator (1-hot) vectors, as a Bernoulli variable with rate $\sigma(\theta)$. $\sigma = 1$ corresponds to the zero-noise condition, and with rate $1 - \sigma(\theta)$, a label is dropped out. For this noise, the information about each label is $I[y_\theta \mid x_\theta] = 1 - H_b(\sigma(\theta))$, where $H_b(\sigma)$ is the binary entropy function. Together, the rate of Bernoulli noise is given by $\sigma(\theta) = H_b^{-1}(1 - \frac{a}{p(\theta)})$. We approximated the inverse binary entropy function with a table lookup and assuming $\sigma \geq 0.5$.

Sensitivity analysis of linear networks

In this section we will analyze the sensitivity of a linear multilayer neural network in which the weights of layer i are parameterized by W_i. The output of such a network with N layers is:

$$Y = W_N W_{N-1} \ldots W_2 W_1 X \quad (2)$$

The product matrix is simply W, such that $Y = WX$.

Throughout this analysis we will make heavy use of the singular value decomposition of the weight matrix, which defines matrices U, S, and V such that $W = USV^T$. The matrix S is diagonal, and the diagonal elements are called the singular values σ_i. The U and V matrices are orthonormal.

Our analysis describes how learning dynamics in this system acts to link output sensitivity to input statistics. Note that the derivation here is for arbitrary objectives; the instance of a reconstruction loss discussed in the main text is a special case. The analysis is organized in three stages: 1) how the sensitivity depends on the singular values σ_i of W, 2) how σ_i change with learning, and 3) how σ_i correspond to the image statistics.

Sensitivity depends on the singular values of *W*

We wish to derive the sensitivity of a linear network to arbitrary input features. We will first examine the case of determining the sensitivity of the network the following feature: how much the data aligns with each jth singular vector of W. This is a weight-dependent feature. Specifically, let the feature θ_j be the dot product of the data with the jth right singular vector of the weight product matrix, $\theta_j = V_j^T x$. This feature is important as it can be used to analyze the sensitivity to arbitrary features.

For this feature, we find the sensitivity of the network is $D(Y\,;\,\theta_j) = \sigma_j^2$. This result is intuitive, as σ_j^2 describes how much data lying along the vector v_j is amplified when multiplied with W. A derivation can be found in the Appendix. Thus, when $\theta_j = V_j^T x$, the sensitivity $D(Y\,;\,\theta_j)$ is constant and is the square of the associated singular value.

The sensitivity to more general θ can be understood using this result. This is because the key derivative can be decomposed into the derivatives with respect to right singular vectors: $\frac{\partial y_\mu}{\partial \theta} = \frac{\partial y_\mu}{\partial V^T x}^T \frac{\partial V^T x}{\partial \theta}$. In this case, we find that $D(y;\,\theta) = \sum_j \sigma_j^2 \frac{\partial V_j^T x}{\partial \theta}^2$ (see Appendix for derivation). Thus, for arbitrary θ, the sensitivity depends on how $V_j^T x$ depends on θ times the size of the associated singular value, summed over components j.

The behavior of the singular values

Previous literature describes how the weight matrix changes due to gradient descent (Arora et al, 2019; Saxe et al, 2013). More information about these results can be found in the Appendix.

We first define a (potentially data-dependent) cost function: $\ell(W_N W_{N-1} \ldots W_2 W_1)$

As described by Arora et al (2019), under certain conditions on the weight initialization the direction of the unit vectors u and v rotate with learning in a specific way. Note that they remain unit length during learning. This result, quoted in the Mathematical Appendix as Theorem 2, states that the vectors are static when they align with the singular vectors of the gradient of the loss, $\nabla\ell(W)$. More specifically, if the singular vectors are static then $U^T\nabla\ell(W)V$ is diagonal. This will become an important condition for tying the input statistics to the singular values of W.

Another important result from previous literature describes how singular values of the product matrix W evolve as a function of time t:

$$\begin{aligned}\dot{\sigma}_i(t) &= -N\sigma_i(t)^{\frac{2(N-1)}{N}}\left\langle \nabla_W \ell(W(t)), u_i(t)v_i^T(t)\right\rangle \\ &= -N\sigma_i(t)^{\frac{2(N-1)}{N}} u_i^T(t)\nabla_W\ell(W(t))v_i(t\end{aligned}$$

Thus, each singular value evolves as a product of a function its current size and the network depth multiplied by how much the gradient correlates with the corresponding singular vectors. This formalism assumes continuous learning dynamics; see (Gidel et al, 2019) for a treatment of finite step sizes.

Relation of frequency to sensitivity

In this section we wish to show how the input statistics affect the singular vectors and values of W. Our approach is to show that frequency $p(\theta)$ reflects in the covariance of θ. The covariance affects the rate of learning of the singular values of the weight matrix W.

Frequency vs. variance

In our analysis of how the statistics of data affect sensitivity, we focus on the variance of features. Since previous literature in psychophysics focus on frequency as defined by the vector $p(\theta)$ with a scalar value for each orientation $p(\theta = \theta_j)$ (e.g. Wei and Stocker (2015)), it is appropriate to discuss their relation. Our analysis focuses on variance in part because attributes like orientation measured can occur with a real-valued strength in each image patch when measured by e.g. Gabor filters or Fourier decomposition. Thus a description of $p(\theta)$ in natural images might be more completely characterized with a two-dimensional matrix with dimensions for angle and intensity. Variance summarizes the intensity axis and characterizes how unpredictable each orientation is within each image patch. The second reason we work with variance is that it cleanly relates to the speed of learning.

When features are binary and either present or not, variance and frequency are closely related. Modeling presence as a Bernoulli variable, the frequency is the probability $p(\theta_j)$ and the variance is $\sigma(\theta_j) = p(\theta_j)(1 - p(\theta_j))$. Note that at very small values of $p(\theta_j)$, $\sigma(\theta_j) \sim p(\theta_j)$. However, features that are nearly always present ($p(\theta_j)$ near 1) have a very low variance. It is interesting to note that this behavior aligns with the expectation that efficient sensory systems should dedicate more resources to features whose presence is uncertain ($p(\theta_j) = .5$) than to those whose presence is guaranteed ($p(\theta_j) = 1$). Variance is thus very similar to absolute frequency for rare Bernoulli variables and in general may be a more intuitive measure of feature importance in regards to what determines efficient patterns of sensitivity.

What W learns: autoencoding objective

Further describing the growth of singular values requires a choice of objective. The base case of our study is the autoencoding objective defined for a set of inputs X:

$$\ell(W) = \frac{1}{N}\sum_{i}^{N}(x_i - Wx_i)^T(x_i - Wx_i$$

Our goal is to determine how W evolves for this cost function. We will examine both the singular vectors and the singular values.

During learning, the singular vectors rotate (recall they are unit length and orthogonal) until they reach a fixed point. For this cost function, it is easy to verify that a fixed point of dynamics is when the singular vectors are equal to the principal components of the inputs (see Appendix for proof). That is, the vectors are static when $\Sigma_{xx} = V\Lambda V^T$ and $W = VSV^T$ for the same V but potentially different Λ and S. This alignment is especially relevant given the expression for network sensitivity derived above. With the vectors aligned, the sensitivity to each corresponding principal component of the inputs is given by σ_i^2, the squared singular value of W.

The evolution of sensitivity is thus governed by the evolution of singular values. The rate of change of σ_i is complicated to calculate because the singular vectors can potentially rotate. However, for the sake of analysis one can examine the case when the singular vectors are initialized at the fixed point mentioned above, as in previous literature (Saxe et al, 2013). In this set of initial conditions, the time-evolution of each singular value of W is given by (Saxe et al, 2013; Arora et al, 2019):

$$\dot{\sigma}_i(t) = N\lambda_i \sigma_i(t)^{\frac{2(N-1)}{N}} (1 - \sigma_i(t$$

Note that the rate of learning is controlled by λ_i, the standard deviation of the *i*th principal component of the inputs. The term on the right causes $\sigma_i(t)$ to converge to 1 asymptotically, as is expected as the solution of the full-rank reconstruction problem is $W = I$. For deeper networks ($N \geq 2$), the growth is sigmoidal and approaches a step function as $N \rightarrow \infty$ (see Gidel et al (2019)). Thus, in this axisaligned initialization, the singular values $\sigma_i(t)$ are learned in order of the variance of the associated principal components of the inputs.

Together, these results mean that the sensitivity of a linear network's output to the principal components of the inputs evolve in order of variance when trained on input reconstruction. This is exactly the case for the axis-aligned initialization and approximately true for small initializations. For the single-matrix network displayed in the figure in the main text, the sensitivity to the *j*th PC thus evolves over time *t* as:

$$\dot{D}(Y;\ PC_j)(t) = 2\lambda_j \sqrt{D(Y;\ PC_j)(t)} \left(1 - \sqrt{D(Y;\ PC_j)(t)}\right)$$

What W learns: supervised learning

We can also determine how input statistics affect the sensitivity for the more general class of objective functions when *Wx* is trained to match some target *y* by minimizing the mean-squared error:

$$\ell(W) = \sum_j (y_j - Wx_j)^2$$

As before, we can gain intuition about W by beginning from an initialization that is axis-aligned with the final solution. For the supervised case, these initializations share the singular vectors of the data/labels, but can differ in the singular values.

Given $\Sigma_{xx} = V\Lambda V^T$ and $\Sigma_{xy} = UTV^T$, we set $W(0) = USV^T$ for the same U and V. See the Appendix for proof that this is a fixed point of singular vector dynamics.

This initialization allows us to understand how the singular values of the weight matrix change. As derived in the Appendix, the time evolution of σ_i is given by:

$$\begin{aligned}\dot{\sigma}_i(t) &= -N\sigma_i(t)^{\frac{2(N-1)}{N}}(\sigma_i(t)\lambda_i - t_i)\\ &= \lambda_i N\sigma_i(t)^{\frac{2N-1}{N}}(\frac{t_i}{\lambda_i} - \sigma_i(t))\end{aligned}$$

As in the case for input reconstruction, the *i*th singular value approaches a target. Instead of 1, this value is $\frac{t_i}{\lambda_i}$, the ratio of the importance of this component (the input/output singular value t_i) and the standard deviation of that component in the inputs λ_i. The growth rate is controlled by the distance from this asymptote (right term) and as well as on the input statistics λ_i. Thus, even for the case of supervised learning the input statistics affect what is learned first via gradient descent directly through Σ_{xx} via λ_i, and not just through the input/label covariance Σ_{xy}.

References

Ahissar M, Hochstein S (1997) Task difficulty and the specificity of perceptual learning. Nature 387(6631):401–406. ISBN: 1476-4687 Publisher: Nature Publishing Group

Appelle S (1972) Perception and discrimination as a function of stimulus orientation: the" oblique effect" in man and animals. Psychological bulletin 78(4):266. ISBN: 1939-1455 Publisher: American Psychological Association

Armstrong V, Maurer D, Lewis TL (2009) Sensitivity to first-and second-order motion and form in children and adults. Vision Research 49(23):2774–2781. ISBN: 0042-6989 Publisher: Elsevier

Arora S, Cohen N, Hu W, et al (2019) Implicit regularization in deep matrix factorization. Advances in Neural Information Processing Systems 32:7413–7424

Banks MS, Crowell JA (1993) Front-end limitations to infant spatial vision: Examination of two analyses. Early visual development: Normal and abnormal pp 91–116. Publisher: Oxford University Press New York

Barlow HB (1961) Possible principles underlying the transformation of sensory messages. Sensory communication 1(01)

Barlow HB (1989) Unsupervised learning. Neural computation 1(3):295–311. ISBN: 0899-7667 Publisher: MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ...

Bell AJ, Sejnowski TJ (1995) An information-maximization approach to blind separation and blind deconvolution. Neural computation 7(6):1129–1159

Benjamin A, Qiu C, Zhang LQ, et al (2019) Shared visual illusions between humans and artificial neural networks. In: 2019 Conference on Cognitive Computational Neuroscience. Cognitive Computational Neuroscience, Berlin, Germany, https://doi.org/10.32470/CCN.2019.1299-0, URL https://ccneuro.org/2019/Papers/ViewPapers.asp?PaperNum=1299

Benucci A, Saleem AB, Carandini M (2013) Adaptation maintains population homeostasis in primary visual cortex. Nature neuroscience 16(6):724–729. ISBN: 1546-1726 Publisher: Nature Publishing Group

Bordelon B, Canatar A, Pehlevan C (2020) Spectrum dependent learning curves in kernel regression and wide neural networks. In: International Conference on Machine Learning. PMLR, pp 1024–1034

Braddick O, Atkinson J (2011) Development of human visual function. Vision research 51(13):1588–1609. ISBN: 0042-6989 Publisher: Elsevier

Brito CS, Gerstner W (2016) Nonlinear Hebbian learning as a unifying principle in receptive field formation. PLoS computational biology 12(9):e1005,070. ISBN: 1553-734X Publisher: Public Library of Science San Francisco, CA USA

Caelli T, Brettel H, Rentschler I, et al (1983) Discrimination thresholds in the twodimensional spatial frequency domain. Vision research 23(2):129–133. ISBN: 00426989 Publisher: Elsevier

Canatar A, Bordelon B, Pehlevan C (2021) Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. Nature communications 12(1):1–12. ISBN: 2041-1723 Publisher: Nature Publishing Group

Coppola DM, Purves HR, McCoy AN, et al (1998) The distribution of oriented contours in the real world. Proceedings of the National Academy of Sciences 95(7):4002–4006. https://doi.org/10.1073/pnas.95.7.4002, URL http://www.pnas.org/cgi/doi/10.1073/pnas.95.7.4002

Deng J, Dong W, Socher R, et al (2009) Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. Ieee, pp 248–255

Dosher BA, Lu ZL (1998) Perceptual learning reflects external noise filtering and internal noise reduction through channel reweighting. Proceedings of the National Academy of Sciences 95(23):13,988–13,993. ISBN: 0027-8424 Publisher: National Acad Sciences

Dosher BA, Lu ZL (2007) The functional form of performance improvements in perceptual learning: learning rates and transfer. Psychological science 18(6):531– 539. ISBN: 0956-7976 Publisher: SAGE Publications Sage CA: Los Angeles, CA

Dosovitskiy A, Beyer L, Kolesnikov A, et al (2020) An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:201011929

Fairhall AL, Lewen GD, Bialek W, et al (2001) Efficiency and ambiguity in an adaptive neural code. Nature 412(6849):787–792. ISBN: 1476-4687 Publisher: Nature Publishing Group

Fechner GT (1948) Elements of psychophysics, 1860. In: Readings in the history of psychology. Century psychology series, Appleton-Century-Crofts, East Norwalk, CT, US, p 206–213, https://doi.org/10.1037/11304-026

Flesch T, Juechems K, Dumbalska T, et al (2022) Orthogonal representations for robust context-dependent task performance in brains and neural networks. Neuron pp S0896–6273(22)00,005–8. https://doi.org/10.1016/j.neuron.2022.01.005

Ganguli D, Simoncelli EP (2010) Implicit encoding of prior probabilities in optimal neural populations. Advances in neural information processing systems 2010:658– 666. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4209846/

Gidel G, Bach F, Lacoste-Julien S (2019) Implicit Regularization of Discrete Gradient Dynamics in Deep Linear Neural Networks URL http://arxiv.org/abs/1904.13262, _eprint: 1904.13262

Girshick AR, Landy MS, Simoncelli EP (2011) Cardinal rules: Visual orientation perception reflects knowledge of environmental statistics. Nature Neuroscience 14(7):926–932. https://doi.org/10.1038/nn.2831, URL http://dx.doi.org/10.1038/ nn.2831, publisher: Nature Publishing Group

Gold J, Bennett PJ, Sekuler AB (1999) Signal but not noise changes with perceptual learning. Nature 402(6758):176–178. ISBN: 1476-4687 Publisher: Nature Publishing Group

Goldt S, Advani MS, Saxe AM, et al (2020) Dynamics of stochastic gradient descent for two-layer neural networks in the teacher–student setup. Journal of Statistical Mechanics: Theory and Experiment 2020(12):124,010. ISBN: 1742-5468 Publisher: IOP Publishing

Gunasekar S, Woodworth B, Bhojanapalli S, et al (2018) Implicit regularization in matrix factorization. 2018 Information Theory and Applications Workshop, ITA 2018 https://doi.org/10.1109/ITA.2018.8503198, iSBN: 9781728101248 eprint: arXiv:1905.13655v3

Güclü U, van Gerven MA (2015) Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. Journal of Neuroscience 35(27):10,005–10,014. ISBN: 0270-6474 Publisher: Soc Neuroscience

He K, Zhang X, Ren S, et al (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778

Henderson M, Serences J (2021) Biased orientation representations can be explained by experience with non-uniform training set statistics. Tech. rep., https://doi. org/10.1101/2020.07.17.209536, URL https://www.biorxiv.org/content/10.1101/ 2020.07.17.209536v3, company: Cold Spring Harbor Laboratory Distributor: Cold Spring Harbor Laboratory Label: Cold Spring Harbor Laboratory Section: New Results Type: article

Hyvärinen A, Oja E (1997) One-unit learning rules for independent component analysis. In Advances in Neural Information Processing Systems Cambridge, MA: MIT (1):480–486

Intrator N, Cooper LN (1992) Objective function formulation of the BCM theory of visual cortical plasticity: Statistical connections, stability conditions. Neural Networks 5(1):3–17. https://doi.org/10.1016/S0893-6080(05)80003-6

Jacot A, Gabriel F, Hongler C (2018) Neural tangent kernel: Convergence and generalization in neural networks. arXiv preprint arXiv:180607572

Karklin Y, Simoncelli E (2011) Efficient coding of natural images with a population of noisy linear-nonlinear neurons. Advances in neural information processing systems 24

Khaligh-Razavi SM, Kriegeskorte N (2014) Deep supervised, but not unsupervised, models may explain IT cortical representation. PLoS computational biology 10(11):e1003,915. ISBN: 1553-734X Publisher: Public Library of Science San Francisco, USA

Kiorpes L, Movshon JA (1998) Peripheral and central factors limiting the development of contrast sensitivity in macaque monkeys. Vision research 38(1):61–70. ISBN: 0042-6989 Publisher: Elsevier

Kohn A, Movshon JA (2004) Adaptation changes the direction tuning of macaque MT neurons. Nature neuroscience 7(7):764–772. ISBN: 1546-1726 Publisher: Nature Publishing Group

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems 25:1097–1105

Leat SJ, Yadav NK, Irving EL (2009) Development of visual acuity and contrast sensitivity in children. Journal of optometry 2(1):19–26. ISBN: 1888-4296 Publisher: Elsevier

Lee J, Xiao L, Schoenholz S, et al (2019) Wide neural networks of any depth evolve as linear models under gradient descent. Advances in neural information processing systems 32

Lee R, Saxe A, McClelland, James (2014) Modeling perceptual learning with deep networks. In: Proceedings of the Annual Meeting of the Cognitive Science Society, issue: 36

Maurer D, Lewis TL, Brent HP, et al (1999) Rapid Improvement in the Acuity of Infants After Visual Input. Science 286(5437):108–110. https://doi.org/10.1126/science.286.5437.108, URL https://www.science.org/doi/abs/10.1126/science.286.5437.108, publisher: American Association for the Advancement of Science

Mayer DL, Dobson V (1982) Visual acuity development in infants and young children, as assessed by operant preferential looking. Vision research 22(9):1141–1151. ISBN: 0042-6989 Publisher: Elsevier

Movshon JA, Kiorpes L (1993) Biological limits on visual development in primates. Early visual development: Normal and abnormal Publisher: Oxford University Press New York

Munakata Y, McClelland JL (2003) Connectionist models of development. Developmental Science 6(4):413–429. https://doi.org/10.1111/1467-7687.00296, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-7687.00296, eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-7687.00296

Neyshabur B, Bhojanapalli S, McAllester D, et al (2017) Exploring generalization in deep learning. arXiv preprint arXiv:170608947

Olshausen BA, Field DJ (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381(6583):607–609. ISBN: 1476-4687 Publisher: Nature Publishing Group

Pedregosa F, Varoquaux G, Gramfort A, et al (2011) Scikit-learn: Machine learning in Python. the Journal of machine Learning research 12:2825–2830. ISBN: 1532-4435 Publisher: JMLR. org

Rao CR (1945) Information and the accuracy attainable in the estimation of statistical parameters. Reson J Sci Educ 20:78–90

Razin N, Cohen N (2020) Implicit regularization in deep learning may not be explainable by norms. arXiv preprint arXiv:200506398

Ruderman D, Bialek W (1993) Statistics of natural images: Scaling in the woods. Advances in neural information processing systems 6

Saxe AM (2015) Deep linear neural networks: A theory of learning in the brain and mind. Stanford University

Saxe AM, McClelland JL, Ganguli S (2013) Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. arXiv preprint arXiv:13126120

Saxe AM, McClelland JL, Ganguli S (2019) A mathematical theory of semantic development in deep neural networks. Proceedings of the National Academy of Sciences 116(23):11,537–11,546. ISBN: 0027-8424 Publisher: National Acad Sciences

Schoups A, Vogels R, Qian N, et al (2001) Practising orientation identification improves orientation coding in V1 neurons. Nature 412(6846):549–553. ISBN: 1476-4687 Publisher: Nature Publishing Group

Schwartz O, Simoncelli EP (2001) Natural signal statistics and sensory gain control. Nature neuroscience 4(8):819–825. ISBN: 1546-1726 Publisher: Nature Publishing Group

Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:14091556

Smith SL, Le QV (2017) A bayesian perspective on generalization and stochastic gradient descent. arXiv preprint arXiv:171006451

Stringer C, Michaelos M, Tsyboulski D, et al (2021) High-precision coding in visual cortex. Cell 184(10):2767–2778.e15. https://doi.org/10.1016/j.cell.2021.03. 042, URL https://www.sciencedirect.com/science/article/pii/S0092867421003731

Teller DY, Movshon JA (1986) Visual development. Vision research 26(9):1483–1506. ISBN: 0042-6989 Publisher: Elsevier

Tishby N, Zaslavsky N (2015) Deep learning and the information bottleneck principle. In: 2015 IEEE Information Theory Workshop (ITW). IEEE, pp 1–5

Wei XX, Stocker AA (2015) A Bayesian observer model constrained by efficient coding can explain 'anti-Bayesian' percepts. Nature Neuroscience 18(10):1509–1517. https://doi.org/10.1038/nn.4105

Wei XX, Stocker AA (2016) Mutual information, Fisher information, and efficient coding. Neural computation 28(2):305–326. ISBN: 0899-7667 Publisher: MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ...

Wei XX, Stocker AA (2017) Lawful relation between perceptual bias and discriminability. Proceedings of the National Academy of Sciences of the United States of America 114(38):10,244–10,249. https://doi.org/10.1073/pnas.1619153114, iSBN: 1619153114

Wenliang LK, Seitz AR (2018) Deep neural networks for modeling visual perceptual learning. Journal of Neuroscience 38(27):6028–6044. https://doi.org/10.1523/JNEUROSCI.1620-17.2018

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE transactions on evolutionary computation 1(1):67–82. ISBN: 1089-778X Publisher: IEEE

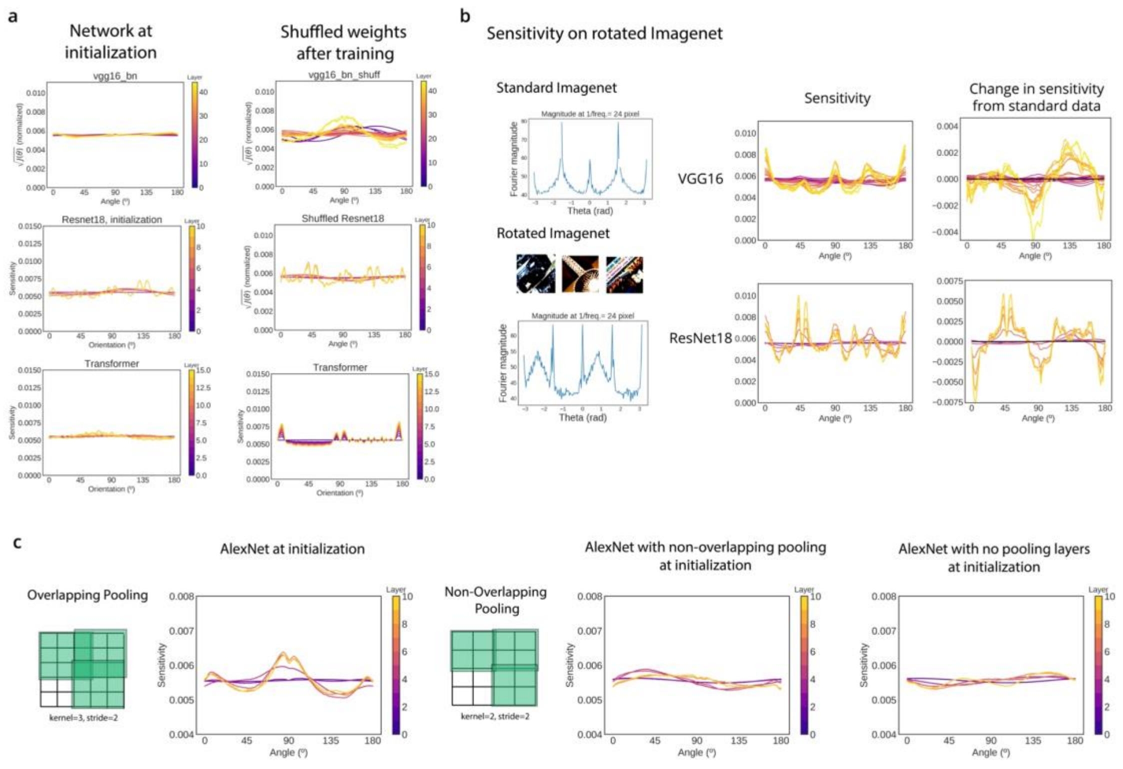
Woodworth B, Gunasekar S, Lee JD, et al (2020) Kernel and rich regimes in overparametrized models. In: Conference on Learning Theory. PMLR, pp 3635–3673

Yamins DL, Hong H, Cadieu CF, et al (2014) Performance-optimized hierarchical models predict neural responses in higher visual cortex. Proceedings of the national academy of sciences 111(23):8619–8624. ISBN: 0027-8424 Publisher: National Acad Sciences

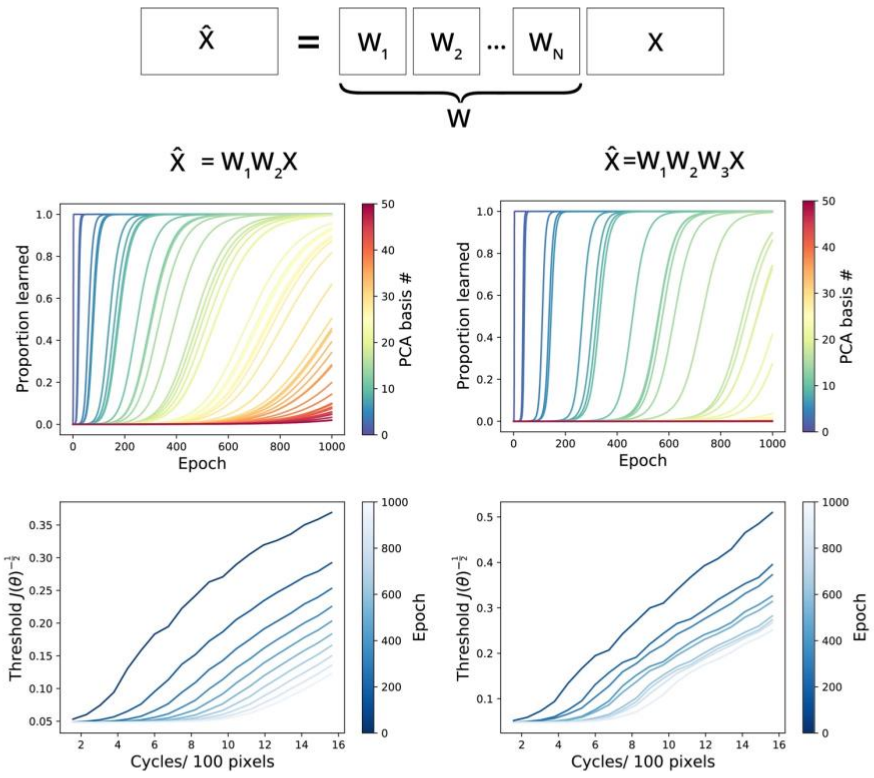
Zhang C, Bengio S, Hardt M, et al (2021) Understanding deep learning (still) requires rethinking generalization. Communications of the ACM 64(3):107–115. ISBN: 0001-0782 Publisher: ACM New York, NY, USA

Zhou S, Yu Y (2018) Synaptic EI balance underlies efficient neural coding. Frontiers in Neuroscience 12:46. ISBN: 1662-453X Publisher: Frontiers

Supplemental figures



SI Fig 1: Controls for orientation sensitivity analyses. a) The sensitivity to the orientation of Gabor stimuli (see Methods for stimuli parameters) at initialization, left column, and after shuffling the parameters within each layer, right column. b) We retrained the Resnet18 and VGG16 architectures on a version of ImageNet in which all images were rotated by 45º. Left: The orientation statistics can be observed as the magnitude of the Fourier spatial decomposition around a circle centered at the origin in frequency space. This method of analysis will show artifacts of spikes at the cardinals due to the edge effects of rectangular images, but it is a useful control that image statistics do change with rotation. Top is for standard ImageNet, and bottom is for rotated ImageNet. Middle column: The sensitivity of ResNet18 and VGG16 after retraining on rotated images. Though changed, it does not appear as a simple shift of the patterns seen for standard ImageNet (Fig. 2). Right: The difference of this sensitivity pattern from the sensitivity pattern observed after training on standard ImageNet shows that changes do correspond to the change in image statistics, at least in part. c) One source of training-independent sensitivity to the cardinal orientations is overlapping pooling, where it is used. (None of the three above networks employ overlapping pooling.) Left: In AlexNet, which does, the network shows a strong nonuniformity of sensitivity at initialization. Right: The use of non-overlapping pooling greatly diminishes the non-uniformity, as does the complete removal of pooling layers (and accompanying change in the convolutional filter downsampling).



SI Fig. 2: Learning dynamics of 2 and 3 layer linear networks. Top: As the depth of networks increases, the principal components are learned in sharper transitions. Bottom: The "threshold" of spatial frequency detection, defined as the inverse square root of the sensitivity to frequency, shows similar patterns regardless of network depth.

Appendix

Relation of sensitivity to the Fisher information

In the main manuscript, we define the *sensitivity* of a data point y to a feature θ as:

$$D(y;\ \theta) = \sum_i \frac{\partial y_i}{\partial \theta}^2$$

Throughout this Appendix, lower-case variables represent vectors and capital letters refer to matrices. We denote the sensitivity an ensemble of data points as $D(Y\,;\theta)$.

The Fisher Information

In this section we show that the above definition of sensitivity can be interpreted as the Fisher Information of y given θ for the case of Gaussian internal noise of unit variance.

Suppose after obtaining y, we observe *noisy* observations of it $\tilde{y}$. Then, the Fisher information of $\tilde{y}$ about some input feature θ is

$$F(\tilde{y};\ \theta) = -\mathbb{E}_{p(\tilde{y}|\theta)}\left[\frac{\partial^2}{\partial \theta^2}\log p(\tilde{y} \mid \theta)\right]$$

When y is a representation of some inputs x, the Fisher information is that implicitly reflects the stimulus ensemble X, through the connection $x \rightarrow \theta \rightarrow y$. It is often the case that we wish to measure the *average* Fisher information over such an input ensemble. For example, the Fisher Information of orientation on natural images, or on test stimuli. We denote such an average by with capital letters, suggesting a matrix of examples: $F(\tilde{Y};\theta)$.

Fisher information for internal noise sources:

We are concerned with the Fisher information $F(\tilde{Y};\ \theta)$ when only a single input x corresponds to each θ. For example, one could be interested in the Fisher information about orientation as characterized by a set of rotating sinusoidal gratings with identical contrast, phase, and frequency.

Since we observe Y noisily, we still observe a distribution $p(\tilde{y} \mid x = f(\theta))$ for each θ. Here we have written $x = f(\theta)$ for some real function $f : \mathbb{R} \to \mathbb{R}$ since x is deterministic given θ. In this case the Fisher information is

$$F(\tilde{Y} \mid \theta) = -\mathbb{E}_{p(\tilde{y}|x=f(\theta))}\left[\frac{\partial^2}{\partial\theta^2}\log p(\tilde{y} \mid x = f(\theta))\right]$$

This is the expression we focus on in this paper. It is simple to calculate using derivatives (as detailed below) and a valid point of comparison to human data characterized with a stimulus ensemble with only one θ per x.

The addition of external noise sources:

As an aside, if one wants to determine the Fisher information about orientation *on natural images*, the desired quantity changes. In naturalistic datasets there are many inputs x that could correspond to each value of θ. Thus, there are two sources of noise: *external* noise producing the distribution $p(x \mid \theta)$, and *internal* noise producing $p(\tilde{y} \mid y)$ or alternatively $p(\tilde{y} \mid x)$. In this case the Fisher information is not easily obtained from derivatives of model representations. This is because this quantity requires marginalizing over x inside of the Fisher expression:

$$F(\tilde{Y} \mid \theta) = -\mathbb{E}_{p(\tilde{y}|x)}\left[\frac{\partial^2}{\partial\theta^2}\log \mathbb{E}_{p(x|\theta)} p(\tilde{y} \mid x)\right]$$

The expectation inside of the log makes this analytically intractable. Approximations may be useful for rather simple $p(x \mid \theta)$, but this is prohibitive for naturalistic data. Thus our expression for sensitivity is not comparable to the Fisher information about data with external noise.

Gaussian internal noise

The notion of network sensitivity in our paper can be equated with Fisher Information in the case that one observes a noised $\tilde{y}$ containing an additive injection of zero-mean Gaussian noise. Thus with noise $\zeta \sim N(0,1)$, $\tilde{y} = y + \zeta$.

In this case $F(\tilde{y}; \theta \mid X)$ simplifies. Since the noise is independent over outputs y_i and furthermore Gaussian over each output unit,

$$\log p(\tilde{y} \mid x) = \sum_i \log p(\tilde{y}_i \mid x)$$
$$\log p(\tilde{y} \mid x) = -\sum_i (\tilde{y}_i - y)^2 - \frac{1}{2}\log 2\pi$$

Taking the derivative and expectation over ζ, we obtain the well-known result for the Fisher of Gaussians:

$$\begin{aligned}
-\mathbb{E}_{p(\tilde{y}|x)} \frac{\partial^2}{\partial\theta^2} \log p(\tilde{y} \mid x) &= \mathbb{E}_\zeta[\sum_i \frac{\partial^2}{\partial\theta^2}(\tilde{y} - y)_i^2] \\
&= -\mathbb{E}_\zeta[\sum_i \frac{\partial}{\partial\theta}((\tilde{y}_i - y)\frac{\partial y}{\partial\theta})_i] \\
&= \mathbb{E}_\zeta[\sum_i (\frac{\partial y}{\partial\theta})_i^2 - \zeta(\frac{\partial^2 y}{\partial\theta^2}])_i] \\
&= \sum_i (\frac{\partial y}{\partial\theta})_i^2
\end{aligned}$$

Sensitivity analysis of a linear network

Imagine that we have a linear multilayer neural network in which the weights of layer i are parameterized by W_i. The output of such a network with N layers is $Y = W_N W_{N-1} \ldots W_2 W_1 X$. The product matrix is simply W, and $Y = WX$.

The total sensitivity can be broken up into terms that depend on the decomposition of W. This will be the bridge to a theory of learning.

As a minimal example, let us first examine the case of determining the sensitivity as a function of how much the data aligns with each singular vector of W. That is, our feature θ_j is the dot product of the data with the jth right singular vector of the weight product matrix. As discussed later, for autoencoding cost functions this will align with the principal components of the data after a bit of training, and so this might also be said to be the sensitivity about the jth principal component.

Defining the SVD and feature of interest as,

$$\begin{aligned} W &= USV^T \\ \theta_j &= V_j^T x \end{aligned}$$

For this feature, the expected sensitivity for an input ensemble $p(x)$ is:

$$\begin{aligned} D(Y;\ \theta_j) &= \mathbb{E}_{p(x)} \sum_i \big(\frac{\partial USV^T x}{\partial V_j^T x}\big)_i^2 \\ &= \mathbb{E}_{p(x)} \big(US\frac{\partial V^T x}{\partial V_j^T x}\big)^T \big(US\frac{\partial V^T x}{\partial V_j^T x}\big) \end{aligned}$$

Since $\frac{\partial V^T x}{\partial V_j^T x}$ is a one-hot vector that is 1 in the *j*th row and zero everywhere else, it acts to "pick out" the *j*th column of US. Note that the expectation over $p(x)$ disappears as well.

$$\begin{aligned} D(Y;\ \theta_j) &= (US)_{:,j}^T (US)_{:,j} \\ &= \sigma_j U_{:,j}^T U_{:,j} \sigma_j \\ &= \sigma_j^2 \end{aligned}$$

Here $U_{:,j}^T U_{:,j} = 1$ because columns of U are orthonormal. Thus, for when $\theta_j = V_j^T x$, the sensitivity $D(Y\,;\,\theta)_j$ is constant and is the square of the associated singular value.

More general θ

More general θ can be understood using the result of the last section. The approach is to decompose an arbitrary derivative into the derivatives with respect to right singular vectors, as such: $\frac{\partial y_\mu}{\partial \theta} = \frac{\partial y_\mu}{\partial V^T x}^T \frac{\partial V^T x}{\partial \theta}$. In this case,

$$
\begin{aligned}
D(y;\ \theta) &= \frac{\partial Wx}{\partial \theta}^T \frac{\partial Wx}{\partial \theta} \\
&= (\frac{\partial USV^Tx}{\partial V^Tx}^T \frac{\partial V^Tx}{\partial \theta})^T (\frac{\partial USV^Tx}{\partial V^Tx}^T \frac{\partial V^Tx}{\partial \theta}) \\
&= \frac{\partial V^Tx}{\partial \theta}^T SU^TUS \frac{\partial V^Tx}{\partial \theta} \\
&= \frac{\partial V^Tx}{\partial \theta}^T S^2 \frac{\partial V^Tx}{\partial \theta} \\
&= \sum_j \sigma_j^2 \frac{\partial V_j^Tx}{\partial \theta}^2
\end{aligned}
$$

Thus, for arbitrary θ, the sensitivity depends on the derivative of the jth right singular vector with respect to θ times the size of its associated singular value.

The behavior of the singular values

Let us now establish a cost function upon the product matrix.

$$\ell(W_N W_{N-1} \ldots W_2 W_1)$$

Results of previous literature

Though many papers have adopted this framework, as cited in the main text, here we quote the result of Arora et al (2019).

Summary: *During gradient descent, under certain restrictive conditions on the initial values of Wi, the singular values of the product matrix evolve qualitatively differently for N = 1 vs. N > 1. For N > 1 they grow larger sigmoidally (roughly one-at-a-time) and in order of their contribution to the cost ℓ(W).*

The results to follow examine what happens when we train W_i via gradient descent to minimize $\ell(W)$. Each W_i now becomes a function of time, $W_i(t)$, and

$$\dot{W}_i(t) = -\frac{\partial}{\partial W_i}\ell(W_N W_{N-1} \ldots W_2 W_1)$$

In addition we assume that the matrices are initialized in a *balanced* manner, meaning that for all $j < N$,

$$W_{j+1}^T(0)W_{j+1}(0) = W_j(0)W_j^T(0)$$

This holds approximately when the weights are initialized very close to zero.

Lemma (Arora et al. 2019)

The product matrix W(t) can be expressed as W(t) = U(t)S(t)V T(t) where U(t) and V (t) have orthonormal columns and S(t) is diagonal.

Our theory hinges on the behavior of the diagonal elements of $S(t)$, which we will denote as $\sigma_i(t)$.

Theorem 1 (Arora et al. 2019)

The singular values of the product matrix W (t) evolve by:

$$\begin{aligned}\dot{\sigma}_i(t) &= -N\sigma_i(t)^{\frac{2(N-1)}{N}}\left\langle \nabla_W \ell(W(t)), u_i(t)v_i^T(t)\right\rangle \\ &= -N\sigma_i(t)^{\frac{2(N-1)}{N}} u_i^T(t)\nabla_W \ell(W(t))v_i(t)\end{aligned}$$

Thus, each singular value evolves as a product of a function its current size and the network depth ($N\sigma_i(t)^{\frac{2(N-1)}{N}}$) multiplied by how much the gradient correlates with the rank-1 matrix implied by the singular vectors. Note that if $N = 1$ there is no dependence on the current size of $\sigma_i(t)$.

Another important result concerns the rotation of the unit vectors $u(t)$ and $v(t)$. It states that the vectors are static when they align with the singular vectors of $\nabla \ell W(t)$.

Theorem 2 (Arora et al. 2019)

Assume that at initialization, the singular values of the product matrix W (t) are distinct from zero, and that the matrix factorization is non-degenerate, i.e. has depth N

≥ 2. Then, for any time t such that the singular vectors of the product matrix W (t) are stationary, i.e. U˙ (t) = 0 and V˙ (t) = 0, then UT(t)∇ℓ(W(t))V (t) is diagonal.

Relation of input statistics to sensitivity

Our approach is to show that frequency $p(\theta)$ reflects in the covariance of θ. This in turn affects the rate of learning of the singular values of the weight matrix W, at least for certain objectives. This connects the sensitivity, or Fisher Information of $\tilde{Y}$, back to the frequency.

The data covariance affects the learning of $\sigma_i(t)$: autoencoding objective

The base case of our study is the autoencoding objective defined for a set of inputs X:

$$\ell(W) = \frac{1}{N}\sum_{i}^{N}(x_i - Wx_i)^T(x_i - Wx_i)$$

Our goal is to determine the evolution of $\sigma_i(t)$ that results from this cost function. First, see that,

$$\nabla_W \ell(W(t)) = -\frac{1}{N}\sum_{i}^{N}(x_i - Wx_i)x_i^T$$

$$= -\frac{1}{N}\sum_{i}^{N}x_i x_i^T + W\frac{1}{N}\sum_{i}^{N}x_i x_i^T$$
$$= W\Sigma - \Sigma$$

Here Σ is the data covariance, assuming X is centered.

In general, the time evolution of each singular value $\sigma_i(t)$ is complicated to calculate because the singular vectors can potentially rotate, i.e. $(\dot{U})(t) \neq 0$. However, for the sake of analysis we can examine a limited case when the direction of the singular vectors is static. This will allow us to obtain an analytic expression for the evolution of the singular values in terms of the data covariance. In particular we will examine the case when the weight matrix is initialized to share right singular vectors (but not singular values) with the data covariance.

By plugging the expression for $\nabla_W \ell(W(t))$ into Theorem 2, it can be seen that if $\Sigma = V \Lambda V^T$ and $W = VSV^T$ for the same V, then

$$\begin{aligned}
U^T(t)\nabla \ell W(t)V(t) &= U^T(t)\nabla W(t)\Sigma - \Sigma V(t) \\
&= U^T(t)U(t)S(t)V^T(t)V(t)\Lambda V^T(t) - V(t)\Lambda V^T(t)V(t) \\
&= S(t)\Lambda - \Lambda
\end{aligned}$$

This is diagonal, and thus by Theorem 2 $\dot{V}(t) = 0$ during gradient descent on the autoencoding objective.

By Theorem 1, this initialization results in:

$$\begin{aligned}
\dot{\sigma}_i(t) &= -N\sigma_i(t)^{\frac{2(N-1)}{N}}(\sigma_i(t)\lambda_i - \lambda_i) \\
&= N\lambda_i\sigma_i(t)^{\frac{2(N-1)}{N}}(1 - \sigma_i(t))
\end{aligned}$$

Extension to supervised learning

A more general class of objective functions is when W_x is trained to match some target y. The input statistics are again relevant here. If we again take the mean-squared error as the objective,

$$\ell(W) = \sum_j (y_j - Wx_j)^2$$

The evolution of the singular values is determined by the gradient,

$$\begin{aligned}\nabla_W \ell(W) &= \sum_j (y - Wx)x^T \\ &= W \sum_j x_j x_j^T - \sum_j y_j x_j^T \\ &= W\Sigma_{xx} - \Sigma_{xy}\end{aligned}$$

By Theorem 1, then, we have that,

$$\begin{aligned}\sigma_i(t) &= -N\sigma_i(t)^{\frac{2(N-1)}{N}} u_i^T(t)(W(t)\Sigma_{xx} - \Sigma_{xy})v_i(t) \\ &= -N\sigma_i(t)^{\frac{2(N-1)}{N}} u_i^T(t)W(t)\Sigma_{xx}v_i(t) + N\sigma_i(t)^{\frac{2(N-1)}{N}} u_i^T(t)\Sigma_{xy}v_i(t)\end{aligned}$$

Thus, the evolution of the singular values depends on two additive terms. One of these (left) has no dependence on the labels *y*, only on the statistics of the data.

As before, we can gain intuition about this evolution by beginning from an initialization that is axis-aligned with the final solution. For the supervised case, these initializations share the singular vectors of the data/labels, but can differ in the singular values. Given $\Sigma_{xx} = V\Lambda V^T$ and $\Sigma_{xy} = UTV^T$, we set $W(0) = USV^T$ for the same U and V. This means that,

$$U^T(t)\nabla\ell W(t)V(t) = S(t)\Lambda - T$$

This is a diagonal matrix, and thus a fixed point of learning. For this initialization, then,

$$\begin{aligned}\dot{\sigma}_i(t) &= -N\sigma_i(t)^{\frac{2(N-1)}{N}}(\sigma_i(t)\lambda_i - t_i) \\ &= \lambda_i N\sigma_i(t)^{\frac{2N-1}{N}}(\frac{t_i}{\lambda_i} - \sigma_i(t))\end{aligned}$$

Chapter 8: Conclusions

This dissertation aims to justify and implement a machine learning framework for computational neuroscience. Together, these chapters demonstrate new ways in which machine learning can be used as tools (Chapters 2-4) or as a theories (Chapters 4-7) within neuroscience.

Since each chapter relates to a different subfield, the conclusions and future directions drawn from each one are best directed at each field. A general conclusion follows.

Chapters 2 and 3: neurophysiology

The techniques we developed in these chapters are ways to validate descriptions of neural activity and function. Along with these techniques is a message: such descriptions need to be validated in order to be meaningful.

To illustrate this, we documented a failure of assumptions in a few specific situations. These were narrow, by necessity. In recordings of neurons in area V4 in two macaque monkeys, we found that although neurons have strong hue tuning, these do not capture how hue affects those neurons in general (Chapter 2). Likewise, generalized linear models mischaracterize how macaque M1 cells encode arm direction and velocity (Chapter 3). These findings do not alone indict the entire methodology. They are warnings and demonstrations of the necessity of validating a generalization of experimental findings.

Our demonstrations are not the only papers with similar conclusions. It is not a rare situation that features unvaried by a researcher affect the neural response (and as argued in Chapter 2, it should be expected). Area V1 is a canary for the approach. There, papers have asked about generalization of orientation tuning to natural scenes (David et al., 2004; Touryan et al., 2005), and compared typical models with machine learning benchmarks (Cadena et al., 2019; Prenger et al., 2004). As warned by Olshausen and Field (Bruno A Olshausen & Field, 2005, 2006), it seems much about the computations performed in V1 still remain to be understood. This warning exists for V1 because, unusually, the validating measurements have been performed.

In other areas, however, it is rare that key validations are presented along with an encoding model or tuning curve. When omitted, it signals a tacit comfort with the possibility that tuning curves will change with context or that encoding models will miss explainable variance. Encouragingly, this trend may be changing. As experimental methods improve and to allow more neurons to be simultaneously recorded, researchers are again using benchmarks to challenge previous assumptions about the meaning of neural activity (e.g. (Musall, Kaufman, Juavinett, Gluf, & Churchland, 2019)).

In addition to more frequent benchmarking, the future directions for neurophysiology might include theories of processing that guide one's assumptions about responses to untested stimuli. Since the sensory systems know a great deal about the external world (Lillicrap & Kording, 2019), models that incorporate naturalistic statistics are perhaps the most promising. Knowing the statistics of the training data allows neurophysiologists a crucial leg up. An older example taking this approach (to great success) is the notion of efficient coding (Barlow, 1961). Another possibility is to describe the effects of learning effectively with limited exposure to the natural world, as I explored in Chapter 7. By studying the principals by which neural codes emerge, neurophysiology might begin to better understand the codes themselves.

Chapter 5: Probabilistic representation learning

This chapter aimed to further theories of learning. Adopting a popular hypothesis of the computational goal of learning (probabilistic representation learning), it attempted to draw a line reaching down to the cortical circuits that could implement this learning goal. Two ends thus constrained the project: the known biology, and the desired computation.

If any lesson is to be taken from this section, it is that an adversarial algorithm could, in principal, be implemented by the cortex. It is one way the brain might learn internal models of the world via switching between externally- and internally- driven modes of processing (Honey et al., 2017). If this is the case, it would mean that the brain contains discriminators of the two modes of activity, which we hypothesized could be certain interneuron cell types. These would be identifiable by a plasticity rule that switches sign with the mode switch. This algorithm is not implausible, to the extent it is not yet ruled out by known data, and would allow the cortex to solve the difficult and currently unresolved problems inherent to representation learning.

This project treated with less flexibility the top-level computational goal: probabilistic representation learning, learning internal models of neural activity in sensory areas, and Bayesian inference over those models. In certain communities this framework is quite popular (Fiser, Berkes, Orbán, & Lengyel, 2010; Friston, 2005). Yet this theory is not thoroughly linked to a biological substrate, and as described in this project, it is unclear how this objective could be learned by neural circuits. Previous proposals for learning circuits (e.g. (Friston, 2005; Hinton, Dayan, Frey, & Neal, 1995; Rezende & Gerstner, 2014)) have clear problems, and, candidly, my attempt at resolution involved an uncomfortable level of speculation. Since there is inconclusive evidence at the level of biology, the high-level computational goal must do more work to carry the theory than otherwise. If this, too, is inconclusive, perhaps a new and humble attention should be paid to learning circuits, especially during the sensitive or critical period of sensory plasticity.

Chapter 6: Neural network optimization

Neural networks encode functions. For certain questions of learning, one can bracket away the specific parameters that encode those functions and think about how the overall function changes. For neuroscience, this would amount to looking at changes in behavior instead of synapses, but using the mathematical language of optimization.

This chapter proposed a learning rule for ANNs in which the allowed change in behavior is constant over time. This type of learning rule may be at play for animals, as well. In motor tasks, for example, learning appears dependent on the direction of an error but independent of the magnitude of that error (Fine & Thoroughman, 2006). Thinking about optimization in function space, which is well-appreciated in machine learning, may be a concept that is useful for neuroscience as well.

One possible future application in neuroscience is to extend the framework of Chapter 7 and ask about the *behavioral* consequences of learning with algorithms that incrementally adjust behavior. This is to replace gradient descent in parameter space with that in function space, but ask the same question about the residual effects of learning algorithms. This might allow similar insights, but would be agnostic to the specific mechanism by which the brain adjusts behavior. By circumventing the mechanistic issue of what mediates learning, this perspective may afford a more direct (yet abstract) description of the effects of learning on perception and behavior.

Chapter 7: The sensory consequences of learning algorithms

Neuroscience can gain from 'artiphysiology' of neural networks as a complement to neurophysiology. Would artificial neural networks trained on ImageNet be, like humans, more sensitive to basic visual features that are more common? Finding that the answer was 'yes', this chapter documented how this emerges from the learning mechanism of gradient descent. This was a move to link artiphysiology with deep learning theory. The consequences may resonate back from theory, though deep learning representations, and to a better understanding of the brain.

In the course of this project, it became clear that this was a powerful way of thinking and just the beginning of what is possible. This paper focused on efficient coding, but a learning framework may help explain a constellation of neural phenomena. Already it has been tied to why representations appear low-dimensional (Flesch, Juechems, Dumbalska, Saxe, & Summerfield, 2022) and why certain categories are learned before others (Saxe, McClelland, & Ganguli, 2019). Future research may help to further describe these effects of learning in greater detail.

With the bridge between machine learning theory and neuroscience now open, it is important that new concepts continue to be brought over. Machine learning theory is an emerging discipline. Many of its central issues – like why deep neural networks generalize, or what exactly they prefer to learn first – are still unresolved. These theories must be ported to neuroscience when and if they are found. One important area, in particular, is to incorporate theories that bridge the lazy (kernel) and rich (feature-learning) regimes identified by deep learning theory. These insights will almost certainly help to describe learning in the brain.

Conclusion

The process of science is not unlike a learning algorithm. From limited experience, it must make generalizations about an underlying reality or events in the future. This requires making assumptions. If these are incorrect, one is at danger of overgeneralizing (Chapters 2-3). If the assumptions are appropriate, one can learn effectively (Chapter 6) even though the biases of those assumptions are never totally escapable (Chapter 7). Proceeding in both domains requires identifying and optimizing one's prior beliefs.

As the theory of machine learning progresses, this field will have increasingly more to say about learning in the brain and its consequences. It is important that this bridge remain open. Why can we learn some things, and not others? What determines the function of cortical areas, as plastic as they are? These questions will continue to motivate me and hopefully many others in the coming years.

References

Barlow, H. B. (1961). Possible principles underlying the transformation of sensory messages. *Sensory communication, 1*(01).

Cadena, S. A., Denfield, G. H., Walker, E. Y., Gatys, L. A., Tolias, A. S., Bethge, M., & Ecker, A. S. (2019). Deep convolutional models improve predictions of macaque V1 responses to natural images. *PLoS Computational Biology, 15*(4), e1006897.

David, S. V., Vinje, W. E., & Gallant, J. L. (2004). Natural stimulus statistics alter the receptive field structure of v1 neurons. *Journal of Neuroscience, 24*(31), 6991-7006.

Fiser, J., Berkes, P., Orbán, G., & Lengyel, M. (2010). Statistically optimal perception and learning: from behavior to neural representations. *Trends in cognitive sciences, 14*(3), 119-130.

Flesch, T., Juechems, K., Dumbalska, T., Saxe, A., & Summerfield, C. (2022). Orthogonal representations for robust context-dependent task performance in brains and neural networks. *Neuron*, S0896-6273(0822)00005-00008. doi:10.1016/j.neuron.2022.01.005

Friston, K. (2005). A theory of cortical responses. *Philosophical Transactions of the Royal Society B: Biological Sciences, 360*(1456), 815-836.

Hinton, G. E., Dayan, P., Frey, B. J., & Neal, R. M. (1995). The" wake-sleep" algorithm for unsupervised neural networks. *Science, 268*(5214), 1158-1161.

Lillicrap, T. P., & Kording, K. P. (2019). What does it mean to understand a neural network? *arXiv preprint arXiv:1907.06374*.

Musall, S., Kaufman, M. T., Juavinett, A. L., Gluf, S., & Churchland, A. K. (2019). Single-trial neural dynamics are dominated by richly varied movements. *Nature Neuroscience, 22*(10), 1677-1686.

Olshausen, B. A., & Field, D. J. (2005). How close are we to understanding V1? *Neural computation, 17*(8), 1665-1699.

Olshausen, B. A., & Field, D. J. (2006). What is the other 85 percent of V1 doing. *L. van Hemmen, & T. Sejnowski (Eds.), 23*, 182-211.

Prenger, R., Wu, M. C.-K., David, S. V., & Gallant, J. L. (2004). Nonlinear V1 responses to natural scenes revealed by neural network analysis. *Neural Networks, 17*(5), 663-679.

Rezende, D., & Gerstner, W. (2014). Stochastic variational learning in recurrent spiking networks. *Frontiers in Computational Neuroscience, 8*, 38. doi:10.3389/fncom.2014.00038

Saxe, A. M., McClelland, J. L., & Ganguli, S. (2019). A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences, 116*(23), 11537-11546.

Touryan, J., Felsen, G., & Dan, Y. (2005). Spatial structure of complex cell receptive fields measured with natural images. *Neuron, 45*(5), 781-791.

Appendix: Publications

Appearing as chapters in this dissertation

Preprints

Benjamin, Ari S., Ling-Qi Zhang, Cheng Qiu, Alan Stocker, Konrad Paul Kording. "Efficient neural codes naturally emerge through gradient descent learning" *bioRxiv* (2022) https://doi.org/10.1101/2022.05.11.491548

Benjamin, Ari S., and Konrad P. Kording. "Learning to infer in recurrent biological networks" *arXiv preprint* arXiv:2006.10811 (2021).

Benjamin, Ari S., Pavan Ramkumar, Hugo Fernandes, Matthew A. Smith, and Konrad Paul Kording. "Hue tuning curves in V4 change with visual context." *bioRxiv* (2020): 780478.

Peer reviewed

Benjamin, Ari S., David Rolnick, and Konrad Kording. "Measuring and regularizing networks in function space." in *International Conference on Learning Representations,* (2019)

Benjamin, Ari S., Hugo L. Fernandes, Tucker Tomlinson, Pavan Ramkumar, Chris VerSteeg, Raeed H. Chowdhury, Lee E. Miller, and Konrad P. Kording. "Modern Machine Learning as a Benchmark for Fitting Neural Responses." *Frontiers in computational neuroscience* 12 (2018)

Selected as **Editors' Pick 2021.** https://www.frontiersin.org/research-topics/23482/frontiers-in-computational-neuroscience---editors-pick-2021

Reviews

Glaser, Joshua I.*, **Ari S. Benjamin***, Roozbeh Farhoodi*, and Konrad P. Kording. "The roles of supervised machine learning in systems neuroscience". *Progress in neurobiology*. 2019 Apr 1;175:126-37.

Publications not mentioned in this dissertation

Preprints

Lei, Jordan, **Ari S. Benjamin**, and Konrad P. Kording. "Object Based Attention Through Internal Gating" arXiv preprint arXiv:2106.04540 1(2021).

Peer reviewed

Lange, Richard D., **Ari S. Benjamin**, Ralf M. Haefner*, and Xaq Pitkow*. "Interpolating between sampling and variational inference with infinite stochastic mixtures." *Proceedings of the Association for Uncertainty in Artificial Intelligence* (2022)

Marius't Hart, Bernard, Achakulvisut, …, **Ari S. Benjamin**, …, Gunnar Blohm, Konrad Kording, Megan AK Peters, Athena Akrami, Bradly Alicea et al. "Neuromatch Academy: a 3-week, online summer school in computational neuroscience." *Journal of Open Source Education* 5.49 (2022): 118.

Glaser, Joshua I., **Ari S. Benjamin**, Raeed H. Chowdhury, Matthew G. Perich, Lee E. Miller, and Konrad P. Kording. "Machine learning for neural decoding." *Eneuro* 7, no. 4 (2020).

Shen, Hanfei, Tony Liu, Jesse Cui, Piyush Borole, **Ari Benjamin**, Konrad Kording, and David Issadore. "A web-based automated machine learning platform to analyze liquid biopsy data." *Lab on a Chip* 20, no. 12 (2020): 2166-2174.

Chang, Jeffery, **Ari S. Benjamin**, Benjamin Lansdell, and Konrad Paul Kording. "Augmenting Supervised Learning by Meta-learning Unsupervised Local Rules." *NeurIPS 2019 Workshop Neuro AI* (2019)

Reviews

Kording, Konrad P., **Ari S. Benjamin**, Roozbeh Farhoodi, and Joshua I. Glaser. "The Roles of Machine Learning in Biomedical Science." In *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2017 Symposium*. National Academies Press, 2018.