Fall 10-1-2021

Computational Principles of Multiple-Task Learning in Humans and Artificial Neural Networks

Daniel Benjamin Ehrlich
Yale University Graduate School of Arts and Sciences, dehrlich91@gmail.com

Abstract

Computational Principles of Multiple-Task Learning in Humans and Artificial Neural Networks

Daniel B. Ehrlich

2021

While humans can learn to perform many specific and highly specialized behaviors, perhaps what is most unique about human cognitive capabilities is their capacity to generalize, to share information across contexts and adapt to the myriad problems that can arise in complex environments. While it is possible to imagine agents who learn to deal with each challenge they experience separately, humans instead integrate new situations into the framework of the tasks they have experienced in their life, allowing them to reuse insight and strategies across them. Yet the precise forms of shared representations across tasks, as well as computational principles for how sharing of insight over learning multiple tasks may impact behavior, remain uncertain.

The significant complexity in the problem of cognition capable of generalizing across tasks has been both an inspiration and a significant impediment to building useful and insightful models. The increasing utilization of artificial neural networks (ANN) as a model for cortical computation provides a potent opportunity to identify mechanisms and principles underlying multiple-task learning and performance in the brain. In this work we use ANNs in conjunction with human behavior to explore how a single agent may utilize information across multiple tasks to create high performing and general representations.

First, we present a flexible framework to facilitate training recurrent neural networks (RNN), increasing the ease of training models on tasks of interest. Second, we explore how an ANN model can build shared representations to facilitate performance on a wide variety of delay task problems, as well as how such a joint representation can explain

observed phenomena identified in the firing rates of prefrontal cortical neurons. Third, we analyze human multiple-task learning in two tasks and use ANNs to provide insight into how the structure of representations can give rise to the specific learning patterns and generalization strategies observed in humans.

Overall, we provide computational insight into mechanisms of multiple-task learning and generalization as well as use those findings in conjunction with observed human behavior to constrain possible computational mechanisms employed in cortical circuits.

Computational Principles of Multiple-Task Learning in Humans and Artificial Neural Networks

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

by
Daniel B. Ehrlich

Dissertation Director: John D. Murray

December 2021

Acknowledgments

I am deeply grateful to my advisor, John Murray, for the remarkable support and trust he put into me as a graduate student, working with me to pursue projects across a wide variety of domains that stretched the expertise of the lab. His precision and clarity of thought has been a essential at every stage of my work.

I'd also like to thank my first scientific advisor, Ifat Levy, who not only taught me all the core skills of research science, but also instilled in me a deep respect for studies of human behavior. I'd like to thank my thesis committee – Daeyeol Lee, Alex Kwan and Damon Clark – for their support and advice throughout not only my thesis work, but my qualifying process as well.

I'd like to thank the members of the Murray lab for their insight, inspiration, friendship and support: Josh Burt, Norman Lam, Markus Helmer, Murat Demirtas, Julie Goulet, Max Shinn, Amber Howell, Rachel Cooper, Qinglong Gu, Taku Ito, Daming Li, Hui Liang Peng, Warren Pettine, Frederick Berl, Heraclitos Lefcochilos-Fogelquist.

I'd like to thank Alan Anticevic as well as the members of the Anticevic Lab, in particular Lisa Ji, Yvette Afriyie-Agyemang, Nicole Santamauro, and Genevieve Yang.

I owe a true debt of gratitude to the undergraduates for whom I had the pleasure to act in a mentor capacity, for their enormous efforts but also their inspiring enthusiasm for science: Alex Atanasov, David Brandfonbrener, Vanessa Achoy, Jasmine Stone, Adela DePavia, Amber Hu, and Sam Zheng.

Most importantly I would like to thank my friends and family for their unceasing love and support without whom none of this work could have happened.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

While cognitive and systems neuroscience research predominantly proceeds a single well controlled task at a time, in the real world we must learn how to deal with varied situations with experiences and training constantly being interleaved. For that reason it is vital that we consider mechanisms of acquiring and solving multiple tasks in parallel that might facilitate the observed capacity of humans to deal with these challenges. In this work, we explore the ways in which task optimized artificial neural network models (ANNs) can be used to provide insight into neural computations underlying learning and flexible cognition across multiple tasks.

Despite the potential utility of ANNs as a model for neuro-biological computation, there remain barriers to entry that prevent many neuroscience labs from being able to utilize them efficiently. In Chapter 2 we describe a software package ("PsychRNN") we developed to facilitate research using continuous time recurrent neural network (RNN) models[2]. In order to make training RNNs more accessible we abstracted away the deep learning technicalities such that users can focus on task design and network parameter selections. To increase the applicability to neuroscientific questions we included a wide array of neuro-biologically motivated constraints not commonly available in off-the-shelf neural network training platforms.

Using RNNs we then investigated the relationship between flexible cognition on delay tasks and observed neural representations. Prior research recording single unit activity in macaques performing individual delay tasks identified heterogeneous tuning profiles for the same populations in prefrontal cortex[6, 4, 3]. In Chapter 3, we describe a novel paradigm capable of dissociating potential representations and a putative framework to explain how a single unified representation can enable efficient computation. We find our framework was consistent with measured human behavior as well as a solution identified by our RNN. We further identify how the implementation of our RNN model gives rise to phenomena long observed in prefrontal populations during delay tasks.

In contrast to the majority of studies which utilize fully optimized ANN models, we wished to investigate the suitability of ANN models for explaining human rule learning behavior both across tasks learned independently and within multiple tasks learned contemporaneously. It has long been observed that humans can learn a wide variety of rules from experience including those that require non-linear combinations of features[5]. Despite this, efficient learning can benefit from information sharing across rules[1]. We generate a behavioral paradigm of deterministic rule learning, finding evidence of linear components of learning in human behavior. While learning dynamics in a feedforward ANNs also demonstrate a linear learning component, we find that it is incapable of correctly matching between rule learning rate.

Taken together, we show how RNNs can provide a robust and insightful model for the mechanisms underlying algorithmic level solutions to neuro-behavioral problems of interest.

Bibliography

[1] Rich Caruna. Multitask learning. *Machine Learning*, 28:41–75, 1997. 2

[2] Daniel B Ehrlich, Jasmine T Stone, David Brandfonbrener, Alexander Atanasov, and John D Murray. Psychrnn: An accessible and flexible python package for training recurrent neural network models on cognitive tasks. *eNeuro*, 8(1), 2021. 1

[3] S Funahashi, C J Bruce, and P S Goldman-Rakic. Mnemonic coding of visual space in the monkey's dorsolateral prefrontal cortex. *J Neurophysiol*, 61(2):331–49, Feb 1989. 2

[4] J Quintana and J M Fuster. From perception to action: temporal integrative functions of prefrontal and parietal neurons. *Cereb Cortex*, 9(3):213–21, 1999. 2

[5] Roger N Shepard, Carl I Hovland, and Herbert M Jenkins. Learning and memorization of classifications. *Psychological Monographs: General and Applied*, 75(13):1–42, 1961. 2

[6] J D Wallis, K C Anderson, and E K Miller. Single neurons in prefrontal cortex encode abstract rules. *Nature*, 411(6840):953–6, Jun 2001. 2

Chapter 2

Background

2.1 Multi-Task Learning

While it may seem intuitive that being able to learn tasks in isolation from possible interference and distraction is preferable, observed behavior tends to paint a more complex picture. In humans, learning multiple tasks can both facilitate and inhibit accurate rule learning[7], with the direction and magnitude of the effect impacted by precise choices regarding the task pair[30]. Humans in multi-task environments have been found to share policies between contexts[29], enabling quick adaptation to novel task environments.

Research on learning multiple tasks contemporaneously has a long history in statistical learning[4], with multiple theories regarding how training an agent on more than one task could potentially lead to quicker learning trajectories and more robust task performance. First, learning a second related task can act as implicit data augmentation. The inclusion of additional samples could provide a potential advantage where not enough data exists. Even in cases with sufficient samples implicit data augmentation can increase data-efficiency, or the speed at which learning progresses given a number of trials. Second, an auxiliary task can act to regularize representations, incentivizing the agent to learn representations that can be useful for multiple tasks[36]. This type of general representation can help an

agent deal with novel situations. Third, a model can benefit from ”eavesdropping” on the solution to an alternative problem. In cases where a given intermediate learned feature is more easily acquired in one of the pair of tasks, it is possible for the other task to gain advantage through enabling quicker acquisition of that feature than would be possible in learning the task alone[26].

Neurobiological studies of multi-task behavior and learning can help us understand the source and impact of the mechanisms underlying generalization across tasks. Conjunctive representations similar to those one would expect if humans are forming short term response maps have been identified using electroencephalography (EEG)[16]. In functional magnetic resonance imaging (fMRI) studies of humans performing multi-task experiments have linked the adaptive control of multi-task environments to the prefrontal cortex (PFC) as well as a more general network of frontal and parietal regions of cortex[6].

2.2 Executive Function and the Prefrontal Cortex

2.2.1 What is Executive Function?

Executive function is a term often used to describe a set of associated higher order cognitive processes used to facilitate flexible and adaptive behavior[13]. These include processes related to adapting to novel environments such as learning [7], but also to cognition directed at enabling flexible contextual processing [24]. For this reason executive function has often been associated with reasoned deliberative cognition, and contrasted with reflexive behavior, but executive function need not be consciously applied.

The extent to which executive function, an admittedly broad psychological construct, can be considered unified remains unclear. A factor analysis of individual differences identified three main components underlying behavior on a battery of executive function

tasks (1) set shifting, (2) information monitoring and (3) response inhibition [20]. Despite the identification of separable factors the authors also observed correlation between factors leading to the possible that at least in part they share common mechanisms.

2.2.2 The Role of Prefrontal Cortex in Executive Function

The prefrontal cortex composes roughly 10% of the human cortex, resting in the rostral frontal cortex. Notably, the prefrontal cortex is phylogenetically enlarged in both humans and to a lesser extent our near non-human primate relatives.

One of the earliest associations of the prefrontal cortex with executive functions comes from the famous lesion patient Phineas Gage. Gage suffered an accident in which his left prefrontal cortex was largely destroyed. Despite a generally remarkable recovery it was observed that Gage had sustained a broad array of behavioral deficits related to impulse control and planning.

In more recent years, systematic loss of function studies of patients with cortical lesions have implicated a network of frontal parietal regions as central to enabling executive function [1].

Complimentary work in both humans and animal models, using multiple neural measurements, has further explicated the association of prefrontal cortex and executive control. One productive line of research has been to explore evidence of the sub-components of executive function, such as those used in decision making and working memory.

For example, converging lines of research from functional magentic resonance imaging (fMRI) studies in humans have indicated the computation of values necessary for decision making in prefrontal regions [19, 27, 38]. Further analysis has demonstrated not only that PFC activity is modulated by reward components, but that rather that activity correlates with a common and subjective valuation mechanism necessary to compare alternative options [14, 17].

Working memory implementation has been predominantly identified using in-vivo recordings from primates conducting delay tasks. Multiple research groups have identified task variable selective persistent activity in PFC individual neuron firing rates across a variety of paradigms [11, 22, 31]. In humans, evidence of both persistent motor and stimulus selective working memory has been observed in the BOLD response of PFC [9, 8].

2.3 Artificial Neural Networks in Neuroscience

Artificial neural networks (ANNs) form a class of distributed computational models originally based on the observed neuronal connections in the brain. Each model is made up of many independent units, each performing an independent non-linear computation. Units are connected by a set of weights that determine how much the output of one unit will impact another.

Where early models in neuroscience were traditionally hand-designed to match either a neurobiological or behavioral phenomena of interest, ANNs are usually initialized with random connections between units and then optimized to perform a given task. The most common optimization protocols are all variants of what is called gradient descent, in which the model is evaluated for its current set of weights and then each weight is updated a small step in the direction that would have produced better behavior. Over many iterations of this process the networks can eventual reach connection weights that accurately perform many tasks.

Importantly this adds two important features to ANN models. First, ANN models can be constructed without a prima facie hypothesis of the solution. This is extremely important when you wish to investigate the sensitivity of solutions to task or to generate a model for a task for which no known solution exists. Second, models learned by ANNs are not constrained by the near linearity and low dimensionality in which human intuition,

and therefore past models, have predominantly operated.

2.3.1 Feedforward Neural Networks

The most common structure of ANNs are what are referred to as feedforward. Feedforward ANNs are built such that weights between units cannot form a cycle, causing information flow to only process in one direction through the model.

Due to the intrinsically hierarchical nature of processing in feedforward ANNs, they have been used to represent hierarchical processing of sensory inputs across sensory regions in cortex. The earliest and most influential work demonstrated that task optimized feeforward neural netwokrs were capable of explaining more variance in the activity patterns observed in higher order visual object recognition areas in temporal cortex than had previous models hand built to simulate neural data [34, 35]. In addition the authors observed that as models object recognition accuracy improved their fit to neural data did as well, demonstrating an empirical coupling between task performance and match to neural data.

Following this research other groups have expanded to additional sensory modalities including auditory[15] and olfactory systems, indicating that the utility of task optimized ANNs can expand beyond the visual system.

2.3.2 Recurrent Neural Networks

Recurrent neural networks (RNNs), in contrast to feedforward networks, do contain cycles in their between unit connections such that the output from a unit at one time point can impact activity of a different unit at a future time. For this reason they have been used to investigate many behavioral paradigms that unfold across time. Further because of the potential for reciprocal interactions between units, RNNs are capable of enormous

flexibility in their dynamics allowing researchers the ability to use RNN training as a search through possible dynamic solutions to a given computational or cognitive problem.

While in most machine learning contexts RNNs proceed in discrete steps, in our work we focus on what is referred to as continuous time RNNs (ctRNN). ctRNNs use a time constant parameter, τ, to smoothly adapt the state of the network between time points approximating a smooth dynamical system.

$$\tau \dot{x} = -x + W^{\text{rec}} r + W^{\text{in}} u + b^{\text{rec}} + \eta \tag{2.1}$$

$$r = f(x) = \max(x, 0) \tag{2.2}$$

$$z = W^{\text{out}} r + b^{\text{out}} \tag{2.3}$$

This smoothness across time is one that is observed in cortical dynamics [12, 21] making it a useful constraint for RNN models of neural circuits.

Despite recent advances in computational power, RNN modeling of neural and behavioral phenomena has a long history[39]. Recent increases in the size of trainable networks, has led to an explosion in the types of tasks that researchers can successfully and efficiently model using RNNs.

Researchers have used RNNs to study processes as varied as context dependence in perception [18], interval timing and movement production [23], model based learning [3]. Not only can an RNN learn a single task, but of significant relevance to this work, it has been shown that the same RNN model can learn to perform multiple tasks using shared representations to complete them [37].

Analysis of artificial neural network models has proceeded largely along two paths, (1) attempts to find a unified approach that will reveal insight into an arbitrary model [28] and (2) analysis custom built to draw insight from a specific trained ANN [23, 5].

The first method has generally proceeded through identification of common dynamical features, such as fixed and slow points[28] or through identified population level features such as clustering in the recurrent population[10]. While this method has significant promise, in our work we wish to identify more specific descriptions of the algorithmic basis of our ANN models.

For that reason we predominantly use the second analysis framework using a more empirical and custom tailored set of analyses to tease apart rival algorithmic and implementation hypotheses about a trained ANN. This may involve running a trained ANN in a novel experimental setting, using lesions to parts of the ANN, doing direct analysis on the weights or the activity states of units of the network using any of the statistical tools common to empirical research such as clustering or classification.

In many ways this second approach mirrors the methods traditionally applied in systems neuroscience research, with analyses being custom designed to match the questions being addressed by the project and the constraints of the system being evaluated.

2.4 Comparison to Prior Modeling

Modelling in prefrontal cortex has traditionally been isolated to a specific process of interest, with only a few recent studies trying to tackle the question of a system capable of performing multiple tasks[36]. Among these processes are learning (especially structure learning), working memory and flexible cognition.

Traditionally, working memory modeling has been focused on the issue of stimulus maintenance over delays. For this reason early models have focused on mechanisms to form multi-stable circuits which are capable of maintaining states initiated earlier in a trial [32, 33]. Such models can be expanded to encode both binary and continuous variables[25], but the usage of such stored information is overwhelmingly considered a

separate computational problem and left exogenous of implemented models.

In RNN modeling of working memory there is no enforcement of modularity between working memory and other cognitive processes including planning, the decision process itself or cognitive control. In chapter 4 we explore how such a model actually combines these processes into a single computation rather than attempting to coordinate modules implementing each one separately.

Flexible cognition, or the ability to parse stimuli and states into many possible groupings, is another process associated with the PFC. One recent theory involves the random and broadly mixed combination of features for potential further computation[24]. Despite this there has been recent doubt cast on the extent to which such a full and random mix is optimal for cognition[2]. In chapter 4 we investigate a mixed but structured representation for flexible computation in a specific delay task paradigm. In chapter 5 we use largely random mixed models to investigate possible learning dynamics over different deterministic rules, although we find the specific structure of mixing to be important for match to human behavior.

Bibliography

[1] Aron K Barbey, Roberto Colom, Jeffrey Solomon, Frank Krueger, Chad Forbes, and Jordan Grafman. An integrative architecture for general intelligence and executive function revealed by lesion mapping. *Brain*, 135(Pt 4):1154–64, Apr 2012. 6

[2] Silvia Bernardi, Marcus K Benna, Mattia Rigotti, Jérôme Munuera, Stefano Fusi, and C Daniel Salzman. The geometry of abstraction in the hippocampus and prefrontal cortex. *Cell*, 183(4):954–967.e21, Nov 2020. 11

[3] Matthew Botvinick, Jane X Wang, Will Dabney, Kevin J Miller, and Zeb Kurth-Nelson. Deep reinforcement learning and its neuroscientific implications. *Neuron*, 107(4):603–616, 08 2020. 9

[4] Rich Caruna. Multitask learning. *Machine Learning*, 28:41–75, 1997. 4

[5] Warasinee Chaisangmongkon, Sruthi K Swaminathan, David J Freedman, and Xiao-Jing Wang. Computing by robust transience: How the fronto-parietal network performs sequential, category-based decisions. *Neuron*, 93(6):1504–1517.e4, Mar 2017. 9

[6] Michael W Cole, Jeremy R Reynolds, Jonathan D Power, Grega Repovs, Alan Anticevic, and Todd S Braver. Multi-task connectivity reveals flexible hubs for adaptive task control. *Nat Neurosci*, 16(9):1348–55, Sep 2013. 5

[7] Anne G E Collins and Michael J Frank. Cognitive control over learning: creating, clustering, and generalizing task-set structure. *Psychol Rev*, 120(1):190–229, Jan 2013. 4, 5

[8] Clayton E. Curtis and Mark D'Esposito. Persistent activity in the prefrontal cortex during working memory. *Trends Cogn Sci*, 7(9):415–423, Sep 2003. 7

[9] Clayton E Curtis, Vikas Y Rao, and Mark D'Esposito. Maintenance of spatial and motor codes during oculomotor delayed response tasks. *J Neurosci*, 24(16):3944–52, Apr 2004. 7

[10] Alexis Dubreuil, Adrian Valente, Manuel Beiran, Francesca Mastrogiuseppe, and Srdjan Ostojic. Complementary roles of dimensionality and population structure in neural computations. *bioRxiv*, 2021. 10

[11] S Funahashi, C J Bruce, and P S Goldman-Rakic. Mnemonic coding of visual space in the monkey's dorsolateral prefrontal cortex. *J Neurophysiol*, 61(2):331–49, Feb 1989. 7

[12] Peiran Gao, Eric Trautmann, Byron Yu, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, and Surya Ganguli. A theory of multineuronal dimensionality, dynamics and measurement. *bioRxiv*, 2017. 9

[13] Sam J Gilbert and Paul W Burgess. Executive function. *Curr Biol*, 18(3):R110–4, Feb 2008. 5

[14] Joseph W Kable and Paul W Glimcher. The neural correlates of subjective value during intertemporal choice. *Nat Neurosci*, 10(12):1625–33, Dec 2007. 6

[15] Alexander J E Kell, Daniel L K Yamins, Erica N Shook, Sam V Norman-Haignere, and Josh H McDermott. A task-optimized neural network replicates human audi-

tory behavior, predicts brain responses, and reveals a cortical processing hierarchy. *Neuron*, 98(3):630–644.e16, 05 2018. 8

[16] Atsushi Kikumoto and Ulrich Mayr. Conjunctive representations that integrate stimuli, responses, and rules are critical for action selection. *Proc Natl Acad Sci U S A*, 117(19):10603–10608, 05 2020. 5

[17] Ifat Levy, Jason Snell, Amy J Nelson, Aldo Rustichini, and Paul W Glimcher. Neural representation of subjective value under risk and ambiguity. *J Neurophysiol*, 103(2):1036–47, Feb 2010. 6

[18] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, Nov 2013. 9

[19] Daniel McNamee, Antonio Rangel, and John P O'Doherty. Category-dependent and category-independent goal-value codes in human ventromedial prefrontal cortex. *Nat Neurosci*, 16(4):479–85, Apr 2013. 6

[20] A Miyake, N P Friedman, M J Emerson, A H Witzki, A Howerter, and T D Wager. The unity and diversity of executive functions and their contributions to complex "frontal lobe" tasks: a latent variable analysis. *Cogn Psychol*, 41(1):49–100, Aug 2000. 6

[21] John D Murray, Alberto Bernacchia, David J Freedman, Ranulfo Romo, Jonathan D Wallis, Xinying Cai, Camillo Padoa-Schioppa, Tatiana Pasternak, Hyojung Seo, Daeyeol Lee, and Xiao-Jing Wang. A hierarchy of intrinsic timescales across primate cortex. *Nat Neurosci*, 17(12):1661–3, Dec 2014. 9

[22] J Quintana and J M Fuster. From perception to action: temporal integrative functions of prefrontal and parietal neurons. *Cereb Cortex*, 9(3):213–21, 1999. 7

[23] Evan D Remington, Devika Narain, Eghbal A Hosseini, and Mehrdad Jazayeri. Flexible sensorimotor computations through rapid reconfiguration of cortical dynamics. *Neuron*, 98(5):1005–1019.e5, 06 2018. 9

[24] Mattia Rigotti, Omri Barak, Melissa R Warden, Xiao-Jing Wang, Nathaniel D Daw, Earl K Miller, and Stefano Fusi. The importance of mixed selectivity in complex cognitive tasks. *Nature*, 497(7451):585–90, May 2013. 5, 11

[25] R Romo, C D Brody, A Hernández, and L Lemus. Neuronal correlates of parametric working memory in the prefrontal cortex. *Nature*, 399(6735):470–3, Jun 1999. 10

[26] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv*, 2017. 5

[27] Caleb E Strait, Tommy C Blanchard, and Benjamin Y Hayden. Reward value comparison via mutual inhibition in ventromedial prefrontal cortex. *Neuron*, 82(6):1357–66, Jun 2014. 6

[28] David Sussillo and Omri Barak. Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Comput*, 25(3):626–49, Mar 2013. 9, 10

[29] Momchil S Tomov, Eric Schulz, and Samuel J Gershman. Multi-task reinforcement learning in humans. *Nat Hum Behav*, Jan 2021. 4

[30] E M Waldron and F G Ashby. The effects of concurrent task interference on category learning: evidence for multiple category learning systems. *Psychon Bull Rev*, 8(1):168–76, Mar 2001. 4

[31] J D Wallis, K C Anderson, and E K Miller. Single neurons in prefrontal cortex encode abstract rules. *Nature*, 411(6840):953–6, Jun 2001. 7

[32] Xiao-Jing Wang. Synaptic reverberation underlying mnemonic persistent activity. *Trends Neurosci*, 24(8):455–463, Aug 2001. 10

[33] Kong-Fatt Wong and Xiao-Jing Wang. A recurrent network mechanism of time integration in perceptual decisions. *J Neurosci*, 26(4):1314–28, Jan 2006. 10

[34] Daniel L. K. Yamins, Ha Hong, Charles F. Cadieu, Ethan A. Solomon, Darren Seibert, and James J. DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23):8619–8624, 2014. 8

[35] Daniel LK Yamins and James J DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356, 2016. 8

[36] Guangyu Robert Yang, Michael W Cole, and Kanaka Rajan. How to study the neural mechanisms of multiple tasks. *Curr Opin Behav Sci*, 29:134–143, Oct 2019. 4, 10

[37] Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nat Neurosci*, 22(2):297–306, 02 2019. 9

[38] Zhihao Zhang, Jennifer Fanning, Daniel B Ehrlich, Wenting Chen, Daeyeol Lee, and Ifat Levy. Distributed neural representation of saliency controlled value and category during anticipation of rewards and punishments. *Nat Commun*, 8(1):1907, 12 2017. 6

[39] D Zipser and R A Andersen. A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature*, 331(6158):679–84, Feb 1988. 9

Chapter 3

PsychRNN: An Accessible and Flexible Python Package for Training Recurrent Neural Network Models on Cognitive Tasks

Daniel Ehrlich, Jasmine Stone*, David Brandfonbrener, Alex Atanasov, John Murray*

Ehrlich, D. B., Stone, J. T., Brandfonbrener, D., Atanasov, A., Murray, J. D. (2021). PsychRNN: An Accessible and Flexible Python Package for Training Recurrent Neural Network Models on Cognitive Tasks. Eneuro, 8(1).

Abstract

Modeling using recurrent neural networks (RNNs) is quickly becoming a popular strategy to probe the mechanisms behind cognitive tasks. RNNs are able to accomplish many popular cognitive tasks and are highly experimentally tractable due to their observability and manipulability. Modeling is a fast, inexpensive way to investigate the abilities and limitations of proposed mechanisms behind cognitive tasks, and can lead to deeper under-

standing of current proposed mechanisms and new mechanistic hypotheses. These insights can in turn guide experimental focus to better distinguish between current proposals.

Here we introduce an accessible and extensible Python package for training RNNs on a variety of cognitive tasks. Our package provides an accessible framework, requiring only Python and NumPy to define cognitive tasks of interest. The deep learning details are abstracted away so that researchers do not need a machine learning or TensorFlow (TF) background to get started modeling RNNs. PsychRNN includes multiple initializations, loss functions and regularizations as well as a framework to easily add more. We implement a variety of neurobiological motivated constraints in order to allow researchers to build circuits more closely parallel to those in cortex. For projects that require additional customization and researchers who have TF knowledge, the TF-based Backend is easily extensible.

The design of PsychRNN further enables novel investigations. Task modularity makes it easy to investigate how parametric variations in task demands affect network solutions. This modularity means that with only one task definition it is easy to vary many different task parameters iteratively, making such exploration more efficient. Modularity also allowed us to implement task shaping, or curriculum learning, into PsychRNN, with training tasks being adjusted in closed loop based on performance. Experimentalists regularly train animals using shaping tasks—PsychRNN allows investigation of how shaping trajectory choice could affect the observed results.

Modularity also allowed us to implement task shaping, or curriculum learning, into PsychRNN, with training tasks being adjusted in closed loop based on performance. Experimentalists regularly train animals to perform tasks by shaping tasks—PsychRNN allows investigation of how shaping trajectory choice could affect the observed results.

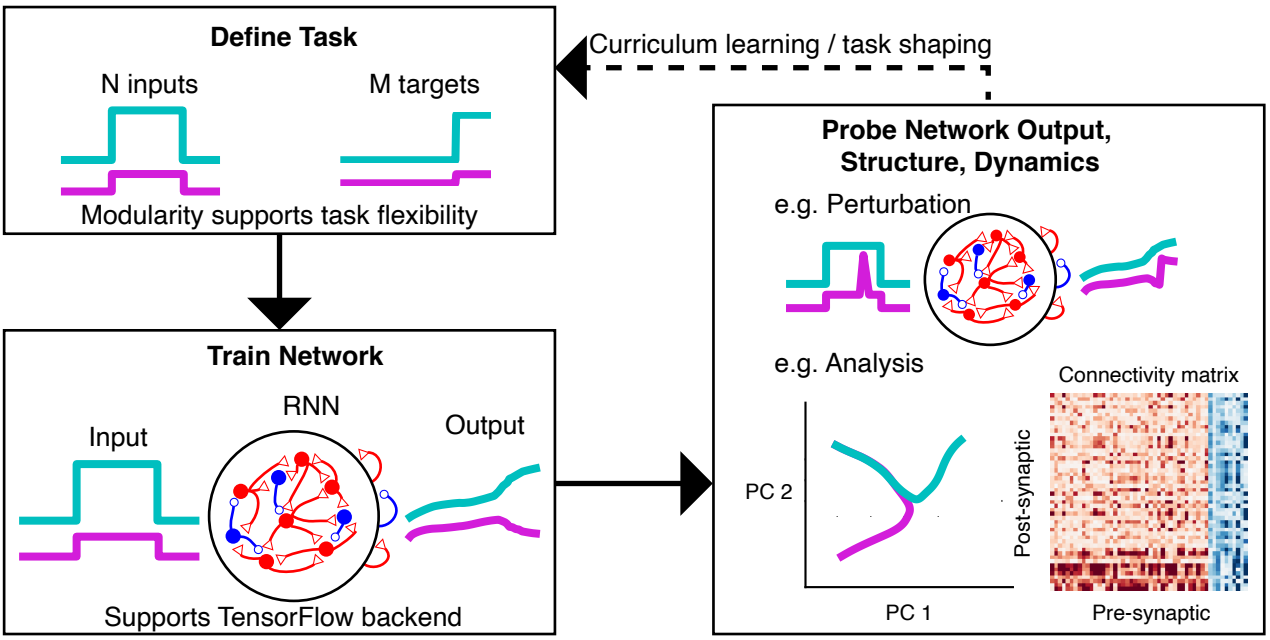


Figure 3.1: **Visual Abstract** Example workflow for using PsychRNN. First one defines the task that one is interested in investigating, and then one trains a potentially biologically-constrained recurrent neural network on that task. Once the network is trained, the dynamics can be perturbed and analyses (e.g. PCA) can be run. The dotted line shows the possible repetition of this cycle with one network by using curriculum learning, also known as task shaping to train the network on a progression of different tasks.

3.1 Introduction

Neural networks trained to perform a given cognitive task successfully simulate neurons and predict neuronal responses in cortex for animal models performing the same task [22, 23]. Multiple research groups have used RNNs in this way to model motor and sensory cortex successfully—they have found that the network dynamics of the RNN align with what we know of neuronal responses [8, 18]. By analyzing the structure of the RNN state variables for a given task, researchers are able to better predict and explain previously not-well-understood features of neuronal responses [8, 12].

Unlike traditional hand-built models, RNNs are ”trained” to perform a given task by updating the weights of the network iteratively over large numbers of training examples. As such they allow researchers to specify the constraints and problem of interest rather than the final model form. The process of allowing the optimization process itself to determine the best algorithmic solution for a given problem makes RNNs a great model for hypothesis generation. Further because solutions are not pre-determined by human researchers, RNNs are less prone to limitations such as near-linearity and low dimensionality that tend to constrain human intuition.

[1] formalized using reverse engineering of RNN solutions to inform our understanding of the brain. Modeling cognitive tasks using RNNs is currently an area of intense interest, as evidenced by the numerous papers concerning this subject [11, 2, 21, 20].

While the basic logic of RNN optimization is straightforward, efficient implementations can be challenging. For this reason, a variety of machine learning frameworks have been released to provide a high level language in which to specify network structure and run optimization. Even with these frameworks, however, getting started with a neuroscientific RNN research program can be challenging. For experimental labs, getting started with machine learning frameworks often requires a dedicated graduate student or postdoc

to learn and be in charge of projects relating to machine learning, a very time-intensive and expensive commitment. Even for theory labs, building a framework for modeling cognitive tasks is time intensive. And, if theorists choose to code each experiment separately rather than developing a framework, significant code is duplicated, and the resulting code files are often hard to understand for others. Further the most powerful and popular machine learning frameworks were designed for general use, and are not specifically tailored to the use case of cognitive- and neuro-scientists.

We thus introduce a framework, PsychRNN, designed to be easy to use and flexible for researchers with any level of deep learning expertise. PsychRNN reduces the time and effort needed to start modeling cognitive tasks using RNNs and thus makes incorporating this research into normal research workflows very easy and convenient. Further PsychRNN implements a variety of features, such as biological connectivity constraints, that may be of special importance to the neuroscience community. Here we introduce the features of PsychRNN, compare it to alternatives, and describe the overall package structure.

PsychRNN is based on the TensorFlow framework and is compatible with both TensorFlow 1 and TensorFlow 2. Importantly TensorFlow is actively supported with features being added on a continuing basis where previous RNN training frameworks for neuroscientists have been built on frameworks since retired [16].

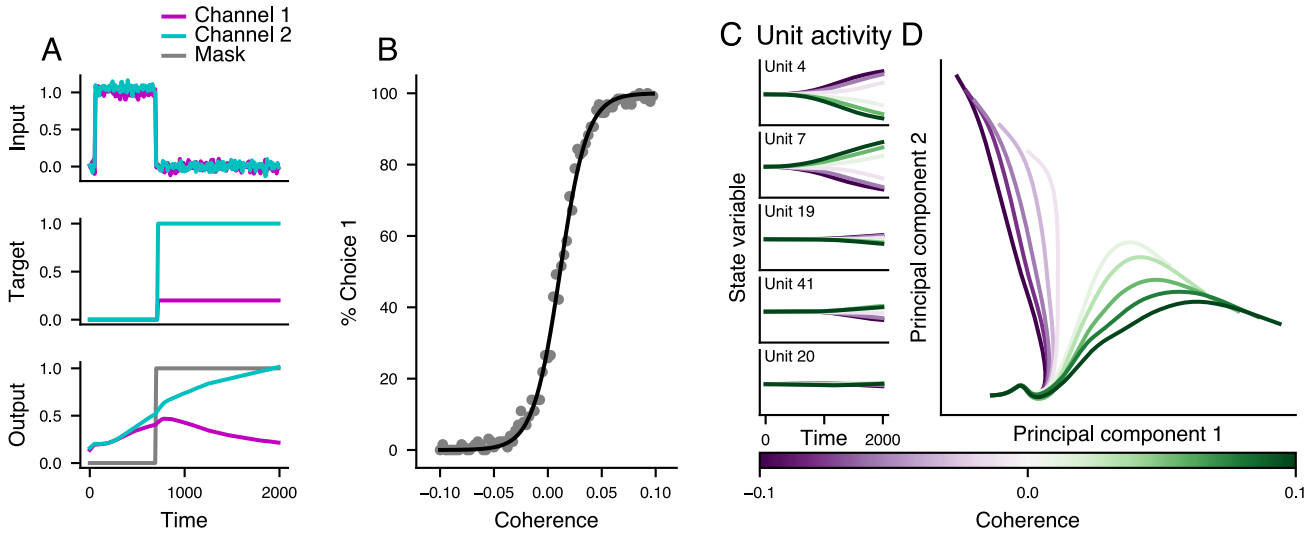
3.2 Results

PsychRNN helps users train continuous time recurrent neural networks (RNNs) on a variety of tasks of interest. We provide a machine learning Backend which converts tasks into input acceptable for a machine learning model, defines such a model in the TensorFlow deep learning framework, and optimizes the network weights using stochastic gradient

descent algorithms. This allows PsychRNN users to focus research resources on the questions of interest rather than the implementation details of an RNN training codebase. As an example, we demonstrate how PsychRNN can specify an RNN model (fig 3.2E), train it to perform a task of neuropsychological interest (here a perceptual discrimination task) (fig 3.2A,B), and return behavioral (the activity of output nodes) and representational outputs (the activity of hidden states) (fig 3.2C,D) with just 11 lines of code (including imports).

The PsychRNN Backend is complimented by the Task Object which enables users to easily and flexibly specify tasks of interest without any pre-requisite knowledge of TensorFlow or machine learning (fig 3.S1). The Task Object allows flexible input and output structure, with tasks varying in not only the temporal features but also the number of input and output channels. Further the object oriented structure of task definition in PsychRNN facilitates tasks that can be quickly and easily varied along multiple dimensions. For example in an implementation of the Delayed Discrimination task we can vary stimuli and delay durations with a set of two parameters (fig 3.3B). Importantly not only can we vary the inputs as they exist, but integration between the Task Object and Backend makes it possible to vary the structure of the network from the Task Object. In our implementation of the Delayed Match to Category task we can freely change the number of inputs (input discretization) and the number of outputs (categories) (fig 3.3D). This flexibility allows researchers to investigate how the network solution of trained RNNs may depend on task or structural features.

While there are substantial efforts elsewhere to create frameworks to make training artificial neural networks accessible, neuroscientific models often require biologically-inspired constraints not common in general purpose RNN software. For this reason, in PsychRNN we include a variety of easily implemented biologically motivated constraints. It is commonly observed that biological neural networks do not adhere to the all-to-all connectivity often used to initialize artificial RNNs. In PsychRNN we give users the option



E

```
1   from psychrnn.tasks.perceptual_discrimination import PerceptualDiscrimination
2   from psychrnn.backend.models.basic import Basic
3   import tensorflow as tf

4   pdm = PerceptualDiscrimination(dt = 10, tau = 100, T = 2000, N_batch = 128)

5   network_params = pdm.get_task_params() # get the params passed in and defined in pdm
6   network_params['name'] = 'model' # name the model uniquely if running mult models in unison
7   network_params['N_rec'] = 50 # set the number of recurrent units in the model
8   model = Basic(network_params) # instantiate a basic vanilla RNN
9   model.train(pdm) # train model to perform pdm task

10  x,target_output,mask, trial_params = pdm.get_trial_batch() # get pdm task inputs and outputs
11  model_output, model_state = model.test(x) # run the model on input x
```

Figure 3.2: **Example Task (Perceptual Discrimination)** (A) The input and target output as specified by the task are shown in the top two panels, and the network's output for the input in the top panel is shown in the bottom pannel. Because the output mask is zero during the stimulus period, that is, the RNN is not penalized or rewarded for whatever output occurs during that period, the network is unconstrained during that period. (B) Plot of the percent of decisions the network makes towards towards direction zero at varying coherence levels. Negative coherences levels indicate a task input of abs(coherence) in direction one. A psychometric function is fit to the data (black) as commonly done with data from animals trained on the Random Dot Motion (RDM) task. This plot validates that the network successfully learned the task and that the simplified task representation still capturing the essence of the task. (C) State variable traces output from trials at a given coherence. The network outputs a state variable trace over time for each recurrent unit in the network. In the network shown here, there are fifty units, and so fifty state variables. These state variables can be thought of as neuronal populations. Only data from correct trials is shown here. One thousand trials were performed at each coherence level—state variable values were averaged across these one thousand trials at each coherence level. Note that the teal and orange lines diverges most sharply in both extremes of coherence

Figure 3.2: (Previous page.) (D) Plot of the first two principle components of the state variables shown in (C) at different coherences. PCA vectors were found by concatenating the data shown in each subplot in (B) to make a big matrix that was [number of state variables x [number of coherences * number of time steps in a trial]] and then demeaning and performing PCA. (E) This code sample is a minimal example for using PsychRNN. All relevant modules are imported (lines 1-4), a PerceptualDiscrimination Task Object is initialized (lines 6-11), the basic RNN model is initialized, built, and trained (lines 13-15), output and state variables are extracted (lines 17-18).

to limit autapses (self connections), apply regional connections between different neural populations, constrain the sign of unit outputs to be strictly positive or negative, and to fix sets of weights at a given value during training (fig 3.4). This enables users not only to build models that more closely match biological circuits but also to explore how different biological limitations impact computational solutions.

One important expansion included in PsychRNN, is a native implementation of curriculum learning. Curriculum learning, sometimes referred to as task shaping in the animal training literature, refers to structuring training examples such that the agent learns easier trials or more basic subtasks first. Curriculum learning has been shown to improve artificial neural network training both in training iterations to convergence and in the final loss. In neuroscience, labs adopt a wide variety of different curricula to get animals to perform full experimental tasks. By including curriculum learning we enable researchers to investigate how the curricula they use with animals may impact their behavioral and neural results as well as identify new curricula that may accelerate animal training. Further, curricula can be used more broadly to investigate how learning may be influenced and biased by the sets of tasks an agent has previously encountered. Lastly, curriculum learning can be used to train networks on tasks that may be too complex to be learned without it.

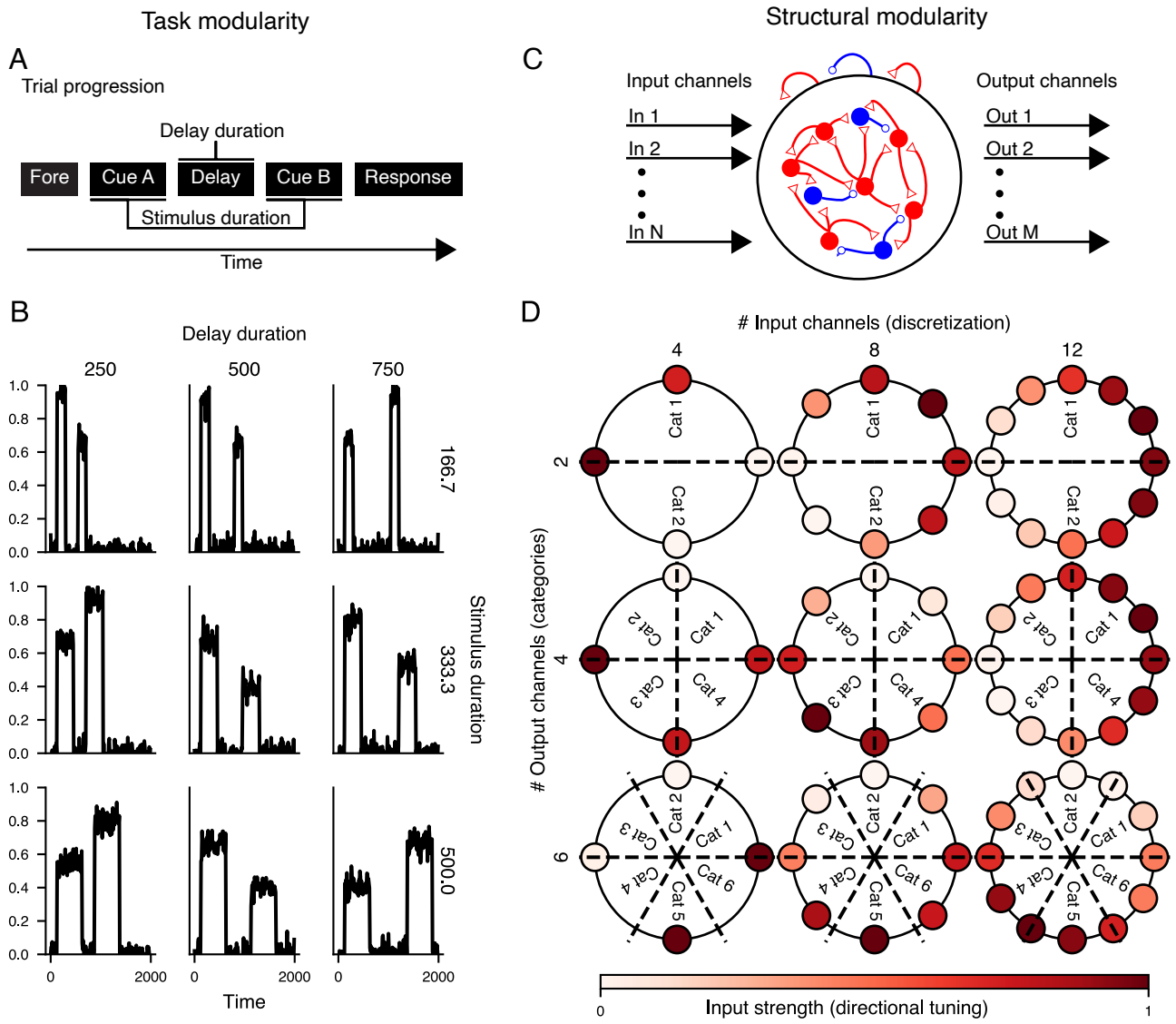


Figure 3.3: **Modularity of Task Definition** (A) Task modularity. This schematic illustrates the trial progression of one trial of a delay discrimination task. The task is modularly defined such that stimulus and delay duration can be varied easily, simply by changing task parameters (fig 3.S2). (B) One channel of input generated by a delay discrimination task with varied stimulus and delay durations. Delay duration is varied across columns, and stimulus duration is varied across rows. In PsychRNN, varying the parameters of the task as illustrated above is easy (fig 3.S3). In previous packages for modeling of cognitive tasks using recurrent neural networks, tasks were not nearly so modular, and varying parameters as seen above would have been much more cumbersome. (C) Structural modularity. Tasks can provide any number of inputs and outputs to train an RNN on (though the number of inputs and outputs should be consistent for any one RNN). (D) Example of a match-to-category task. The number of inputs (colored outer circles) is varied across columns, and the number of output categories (cat) is varied across rows. The variation in number of inputs and outputs was achieved through simple modular task parameters(fig 3.S3).

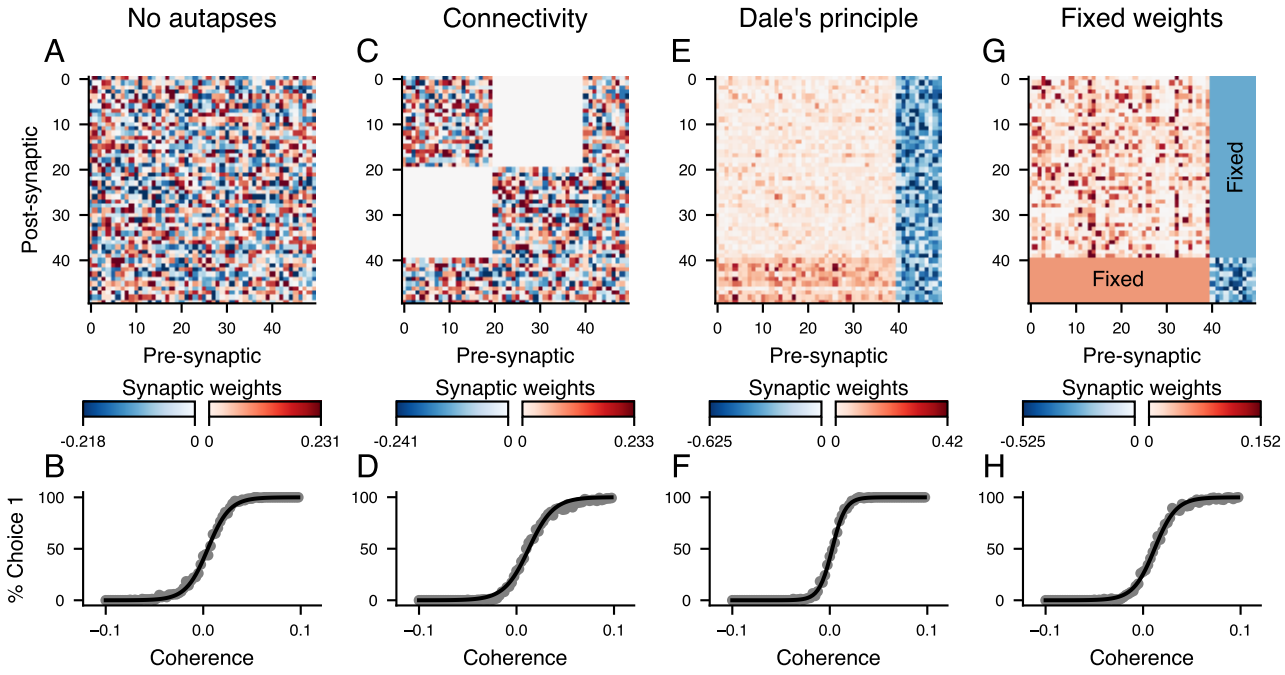


Figure 3.4: **Biologically Motivated Constraints.** This figure illustrates trained recurrent weight matrices and coherence under different biologically motivated constraints. The recurrent weight matrices (A,C,E,G) illustrate the synaptic weights between each of the 50 units in the network. Blue indicates an inhibitory connection and red indicates an excitatory connection. The coherence plots (B,D,F,H) show that the network successfully trains with each of these constraints in place. The network shown in plots a and b is constrained to have no autapses, that is no self connections, as illustrated by the diagonal with weight=0. That shown in c and d is constrained to have two densely connected populations of units with sparse connection between the populations. This can be used to simulate different brain regions. That shown in e and f has Dale's law enforced – each neuron either has entirely inhibitory or entirely excitatory outputs. That shown in g and h has Dale's law enforced and was trained with a subset of weight fixed. In this case, all E-I connections are fixed and only E-E or I-I connections can vary.

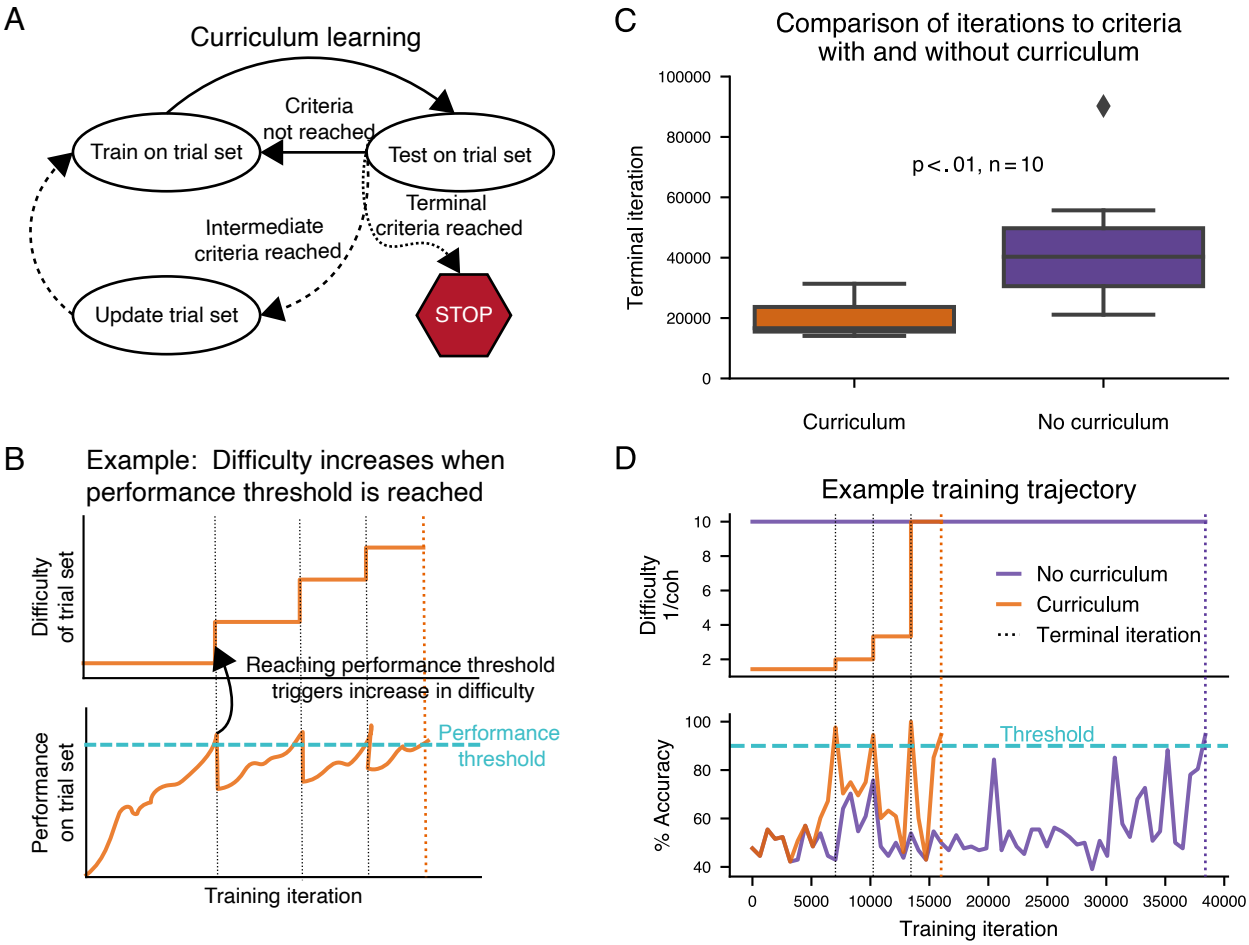


Figure 3.5: **Curriculum Learning** Since PsychRNN includes an easy-to-use implementation of curriculum learning (fig 3.S4), neuroscience researchers can now experiment with curriculum learning with minimal startup costs. (A) In curriculum learning, the network is trained on selections from the trial set, then tested on selections from that trial set. Depending on the performance when testing on the trial set, the trial set can then be updated. This is is similar to the idea of task shaping used in animals – animals are first trained on simpler versions of tasks, and when they get good at those, moved to harder tasks. (B) Schematic of increasing difficulty of trial set (top) paired with performance over time (bottom). The task difficulty is increased when performance reaches the performance threshold. (C) Comparison of number of iterations needed to train a network to perform the perceptual discrimination task at .1 coherence with 90% accuracy. 10 networks were randomly initialized and each was trained both on a curriculum with decreasing coherence, and without a curricula with fixed coherence. Networks trained without curriculum learning were trained solely on stimulus with coherence = .1. Networks trained with curriculum learning were trained with the curriculum shown in fig 3.S4, with coherence decreasing from .7 to .5 to .3 to .1 as performance improved. When the network reached 90 percent accuracy on stimuli with coherence = .1, training was stopped. Networks trained with a curricula reached 90% accuracy significantly faster ($p < .01$).

Figure 3.5: (Previous page.) (D) Difficulty ($\frac{1}{coherence}$), accuracy, and loss (mean squared error) across training iterations on identically initialized networks, one of which was trained with curriculum learning, and one of which was trained without curriculum learning from (C).

3.3 Discussion

PsychRNN provides a robust and modular package for researchers with varying levels of deep learning experience to train RNN models on tasks of interest. The separation into a Python- and NumPy-based Task Object and a primarily TensorFlow-based Backend expands access to RNN model training without reducing flexibility and power for users who require more control over the precise setup of their networks. Further the modularity of tasks and network elements enables easy investigation of how task and structure affect learned solution in RNNs. Lastly the modularization facilitates curriculum learning which makes optimization more efficient and more directly comparable to animal learning.

PsychRNN's commitment to modular task design makes investigating the relationship between problem structure and computational solution more straightforward and approachable. By smoothly varying parameters of a task it is possible to gain detailed insight into the way in which variables of interest may impact both low level features of the resulting networks as well as algorithmic or state space representation. Similarly, modularity over networks facilitates exploration of how structural constraints may impact the family of available solutions to a given task. The ability to easily toggle structural features on and off in our RNN models, such as Dale's Law, connectivity constraints and sparsity regularization, within identically defined task and learning frameworks, makes exploring the impact of biological constraints on network solutions uncomplicated.

Modularity has opened the door to a straightforward implementation of curriculum learning to facilitate experiments regarding learning dynamics and multi-task networks.

One common feature of animal model research is the use of task shaping, or the breaking down of tasks into smaller more easily learned sub-tasks, to train animals to complete complex behavioral tasks. Due to the inefficiency of running multiple parallel shaping strategies for the same experiment, the impact of specific choices made during animal training is often overlooked. One potential use of curriculum learning in PsychRNN is for investigators interested in identifying possible artifacts related to the task shaping strategies used in prior research. By training tasks either on the task without any explicit shaping or on multiple different shaping pipelines it will be possible to generate hypothesis regarding sources of behavioral and neural results attributable to shaping protocol. This use of curriculum to evaluate shaping strategies further could enable researchers to investigate different shaping strategies in a prospective manner, prior to experiment onset. Shaping could be evaluated in RNNs both in terms of ease of learning (fewer training examples necessary to reach criterion behavior) or in terms of least artifactual impact on final trained network. This could provide researchers with a principled basis on which to select between different training schedules for their animal models.

Lastly, the PsychRNN package provides an easy to use framework that can be applied and transferred between research groups to accelerate collaboration and enhance reproducibility. Where in the current environment research groups need to transfer their entire codebase in order to run a neural network model, in the PsychRNN framework they will be able to transfer just a task or model file and other researchers will be able to build off of it or combine it with their own code written in the PsychRNN framework. The ability to test identically specified models across tasks in different groups, and identically specified tasks across models makes comparing and contrasting results substantially more reliable. Even more crucially, the many choices in defining exactly how an RNN will be set up and trained make precise replication of prior published research difficult even in a field where it should be trivial. The specification of PsychRNN task files and parameter dictionar-

ies that can be made public make reproduction of RNN experiments far easier and more transparent.

PsychRNN was designed to reduce startup costs for neuroscience and cognitive science research groups wishing to initiate lines of RNN research, and to extend access to RNN modeling for groups who otherwise might have been dissuaded by technical limitations. In service of this goal we have created a highly user-friendly, modular and clear framework for task specification, while abstracting away much of the deep learning background necessary to train and run networks. While this modularity was intended primarily as a way to ease access to deep learning methods, it simultaneously provides a new access to research avenues, such as multi-task networks and curriculum learning, and generates a reproducible framework that will aid the use of RNN models in neuroscientific research forward in a cohesive and clear manner.

3.4 Methods

The goal of PsychRNN are (1) to facilitate research groups who may not have their own expertise in deep learning to train RNN models on behavioral tasks of interest, (2) to reduce the start up time for researchers who wish to extend our codebase with their own network modules and (3) to aid labs in producing reproducible RNN experiments. To meet these goals and to provide as much utility as possible to users with different programming experience, we have divided the PsychRNN package into two main portions: the Task Object and the Backend, as illustrated in Figure 3.6.

We anticipate that all PsychRNN users will want to be able to define novel tasks specific to their research domains and questions. The Task Object is therefore fully accessible to users without any TensorFlow or deep learning background. Users familiar with Python and NumPy will be able to fully customize novel tasks. Further such users will be able to

customize their network structure (e.g. number of units, form of nonlinearity, connectivity) through preset options built into the Backend.

For users with greater need for flexibility in network design, the Backend is approachable and customizable. Backend customization may, however, require knowledge of TensorFlow. For those with TensorFlow experience, a modular design to RNN model components makes defining new models, regularizations, loss functions and initializations easy. This modularity facilitates testing hypotheses regarding the impact of specific potential structural constraints on RNN training without having to expend time and resources designing a full RNN codebase.

Task

The Task Object is structured to allow users to define their own new task using Python and NumPy (See Figure 3.6).

To specify a novel task a user will need to define two functions: `generate_trial_params` and `trial_function`.

`generate_trial_params` creates trial specific parameters for the task (e.g. coherence and correct decision direction for the Perceptual Discrimination task). It takes two inputs: the batch number and the trial number. Neither of these variables are used in the task definitions included with PsychRNN, but these variables can be used, for example, to balance trials within a batch, or to modify the parameters generated by batch number. `generate_trial_params` returns a dictionary, `params`, containing all the parameters necessary for input to `trial_function`.

`trial_function` specifies the input, target output, and output mask at a given time `t` given the parameters generated by `generate_trial_params`. An example task definition is shown in fig 3.S1.

Built In Tasks

PsychRNN comes set with three example tasks that are well researched by cognitive neuroscientists: perceptual discrimination, delayed discrimination, and match to category. These tasks highlight possible schemas users can apply to specifying their own tasks and provide tasks with which users can test the effect of differint structural network features.

Example Task: Perceptual Discrimination

The PD task is based on classic 2-alternative forced choice perceptual discrimination tasks like the Random Dot Motion (RDM) task [4]. A noisy stimulus is presented to the agent for a finite period, after which the agent must make a choice to one of two targets [4]. The RDM task is a forced choice task – although dots can move in any direction, their are two directions in which the movement of the coherent dots could be [4].

PsychRNN's PD task implementation includes two channels, each representing evidence towards one of the two choices. The float value of a given input channel at a each time point represents the instantaneous coherence of the motion of dots in the direction of represented by that channel. Since there is noise in the signal, integration over time is required at low coherence levels to determine the direction of motion. After the stimulus period is over, a decision is made—the channel of target output representing the correct decision direction is set to one while the channel of target output representing the incorrect decision remains at zero.

As with many behavioral experiments we can restrict output to specific period. In the PsychRNN PD task definition we include an "output mask" such that behavior prior to the output period is not considered during training. Specification of an output mask in a task definition enables users to control at which time points they would like behavior in their task to be constrained and where the network can freely vary. The PD task inputs, target

outputs, and mask can be seen in Figure 3.2A.

During training, the weights of the RNN are optimized such that the network can correctly produce output that matches the target output wherever the mask is nonzero. This provides three core outputs for each trained network model: (1) The activity of output nodes over time on each trial, (2) the activity of recurrent nodes over time on each trial, and (3) the weights (or synapses) from input to recurrent nodes, between recurrent nodes and from recurrent to output nodes.

The activity of output nodes corresponds with the behavior of the network (Figure 3.2A, bottom panel). This output can be used to determine the behavior on different conditions, patterns of errors or the time course of behavior. In addition to analyzing behavior on the trained task, PsychRNN makes it easy to test networks on trials for which they have not been explicitly trained allowing finer grade insight into the underlying computation.

The activity of the recurrent nodes corresponds to the neural representations underlying task behavior (see Figure 3.2C). Specifically the activity is the instantaneous state of the specific unit prior to application nonlinearity. Analyses of the clustering, dimensionality and geometry of the internal states of the RNN can provide insight into the algorithmic implementation of the computation being performed by the RNN.

Lastly, the synaptic weight matrices enable researchers to investigate how structural properties of different networks contribute to specific activity patterns and behavior, and to identify how different tasks impact structural features of the resulting matrix.

Example Task: Delayed Discrimination

The use of a modular Task Object in PsychRNN aides researchers in investigating how different task parameters may impact network results. Because task classes can be defined such that each instance can be initialized with different parameters, it is easy to search through task space.

This task modularity is exemplified in the Delayed Discrimination (DD) task (see Figure 3.3A for a schematic of the task). The DD task is based on the parametric working memory task used by [15]. In the Vibrotactile DD task, the animal feels a first frequency stimulus and must store a trace of that frequency value for comparison [15]. Then, after a delay, the animal feels a second frequency stimulus and must indicate whether the second stimulus frequency is higher or lower than the first [15]. In PsychRNN's implementation during the stimulus epochs two input channels indicates the "frequency" using the difference in input strength.

By passing different parameters to each task instance as described above, the stimulus duration and delay duration can be varied individually (see Figure 3.3B) with very little effort (see Code Sample 3.S2). Researchers can also experiment with the two stimulus durations separately, onset time, decision duration, or any other task parameters they decide to include in their task definition.

Example Task: Delayed Match to Category

Modularity in task definitions can extend beyond varying the input timings and magnitudes. Task modularity allows varying the structure of inputs and outputs.

The Match to Category task is set up to allow such structural modularity. The Match to Category (MTC) task is based off of tasks in which stimuli smoothly vary in some parameter space and must be divided by some arbitrary boundary (Roy 2010). In the PsychRNN MTC implementation, input is a gaussian bump in a particular direction on a ring, and output is based on the slice of a circle that the mean of the gaussian was centered at.

By passing a different N_in parameter to each MTC instance we can vary the number of input channels that define the input discretization. Similarly by varying the N_out parameter we can increase or decrease the number of output categories that the network

must choose between to correctly perform the task (see Figure 3.3C). Tasks can define as many input and output channels as desired (see Figure 3.3C), and the number of inputs and outputs can be varied modularly with very little effort (see Code Sample 3.S3).

Backend

The Backend includes all of the neural network training and specification details. The backend, while being accessible and customizable, was designed with pre-set defaults sufficient to get started with PsychRNN. The TensorFlow details are abstracted away by the Backend so that researchers are free to work with or without an understanding of TensorFlow. Additionally, since the Backend is internally modular, different components of the Backend can be swapped in and out interchangeably. In this section, modular components of the Backend are described so that researchers who want to get more in-depth with PsychRNN know what tools are available to them. Step 2 of Figure 3.6 illustrates the components of the Backend.

Models

Recurrent neural networks are a large class of neural network architectures that process input over time. In the PsychRNN release, we include a basic RNN (what we've been referring to as an RNN throughout the rest of this paper), and an LSTM model (See Figure 3.6, Step 2). The basic RNN model is governed by the following equations

$$\tau dx = (-x + W_{rec}r + b_{rec} + W_{in}u)dt + \sigma_{rec}\sqrt{2\tau}d\xi$$

$$r = f(x)$$

$$z = W_{out}r + b_{out}$$

.

Where u, x and z are the input, recurrent state and output respectively. W_{in}, W_{rec} and W_{out} are the input, recurrent and output synaptic weight matrices. b_{rec} and b_{out} are constant biases into the recurrent and output nodes. dt is the simulation time-step and τ is the intrinsic timescale of recurrent units. σ_{rec} is a constant to scale recurrent unit noise and $d\xi$ is a gaussian noise process with mean 0 and standard deviation 1. $f(x)$ is a nonlinear transfer function (by default rectified linear).

We also include an implementation of an LSTM, a network that enables longer term memory than is easily attainable with the basic RNN [9]. LSTMs use a separate "cell state" to store information gated by sigmoidal units.

Other models can also be defined by users, but require a strong understanding of TensorFlow.

Initializations

The weights that define a neural network are typically initialized randomly. However, with RNNs, large differences in performance, in training time and total asymptotic loss, have been observed for different initializations [19]. Since initializations can be crucial in ensuring the network trains consistently, we have included a few of the initializations currently used in the field [7]. By default, the recurrent weights are initialized with a gaussian spectral radius of 1.1 [17]. We also include an initialization called Alpha Identity (which can be selected by passing that initializer into the network model when it is instantiated) introduced in [19] that initializes the recurrent weights as an identity matrix scaled by alpha. Each of these initializations make the network much more likely to learn a given task successfully.

PsychRNN includes a `WeightInitialization` class that initializes all network weights randomly, all biases as zero, and connectivity masks as all to all. Any new initial-

izations must inherent this class and can override any variety of initializations defined in the base class `WeightInitialization`.

Loss Functions

The RNN is optimized to minimize the loss, so the choice of loss function can be crucial for determining exactly what the network learns. By default, the loss function is `mean_squared_error`. Our Backend also includes an option for using `binary_cross_entropy` as the loss function. Other loss functions can be easily defined with some TensorFlow knowledge and added to the `LossFunction` class. Loss functions take in the network output (`predictions`), the target output (`y`) and the output mask, and return a float calculated using the TensorFlow graph.

Regularizers

Regularizers are penalties added to the loss function that may help prevent the network from overfitting to the data. We include options for L1 regularization and L2 regularization for the synaptic weights. These will tend to reduce the magnitude of weights, sparsifying the resulting weight matrices. In addition we include L2 regularization on r, the post nonlinearity recurrent unit activity.

Other regularizations can be added easily to the `Regularizer` class with TensorFlow. By default, no regularizations are used. If nonzero values for the variables used by a given regularization are set in the parameters used to instantiate a model, that regularization will be applied.

Biologically Motivated Constraints

The default RNN network has all to all connectivity, and allows units to have both excitatory and inhibitory connections. However, this does not reflect the biology we know.

PsychRNN includes a framework for easily specifying biological constraints on the model. Biological constraints affect the dynamics of the resulting system and the state variables that we see, and so are of direct interest both to experimentalists (especially those interested in specific biological constraints, or features that arise from the structural constraints of the brain) and to theorists, many of whom are interested in the dynamics of the brain.

But the brain is not fully connected. Neurons are not generally thought to have many self-connections, and there are different regions which are densely connected within a given region, and sparsely connected across areas. We include a framework for specifying the connectivity matrix for the input, output, and recurrent layers, allowing researchers to specify which connections can exist (have nonzero weights) and which cannot. Networks with block-like connectivity matrices can be used to model separate brain regions. Connectivity constraints are implemented as part of the initialization. Weights matrices are multiplied element wise with the connectivity matrices. The connectivity matrices are by default all ones, allowing all-to-all connectivity. Defining alternate connectivity matrices forces sparser, structured connectivity in the network (see Figure 3.4.A-D).

Dale's Principle states that a neuron releases the same set of neurotransmitters at each of its synapses [6]. Since neurotransmitters tend to be either excitatory or inhibitory, theorists have taken this to mean that each neuron has exclusively either excitatory or inhibitory synapses [16, 14]. We thus include an optional parameter to be passed to the RNN model when it is initialized called `dale's ratio`. When `dale's ratio` is passed into the network, that ratio of recurrent units are made to have only positive synapses, or connections, with other neurons or recurrent units and one minus that ratio of the recurrent units are made to have only negative synapses, or connections with other neurons or recurrent units (see Figure 3.4E-F). This allows researchers to investigate how the constraints of neurons affect the dynamics of the network. Additionally, researchers who study, for example, inhibitory interneurons, would otherwise see no relevant correlate in the RNN

for what they are studying. With the implementation of Dale's law, the role of inhibitory populations can be easily investigated using PsychRNN.

Some synaptic connections may be held fixed while learning a task, while others vary. In PsychRNN, we include an optional parameter specifying weights to fix, and weights to allow to train. By default, all weights are allowed to train. See Figure 3.4G-H for an example of fixing some weights during training.

We can represent most cognitive tasks using a variety of different neural networks, so, for many researchers, the interesting question is [1]: What network architectures and biological constraints lead to network states that most closely match what we record from neurons in animals? And what insight can we glean from differences we observe between biology and neural networks? Thus, the ability to enforce biological constraints is essential to make this package relevant and useful for researchers, and is a current topic of interest [1].

Curriculum Learning

Curriculum learning—the presentation of training examples structured into discrete blocks sorted by difficulty—is motivated by the observation that people learn best in stages. For example, children first learn to count, then to add numbers that each fit onto one hand, and finally to add larger numbers. Full-fledged addition would be much harder to learn without these initial training steps. In neuroscience, the animal models that experimentalists use are similarly taught in stages. The animal model is first taught an easy version of the task before progressively learning harder versions, eventually learning the goal task (see Figure 3.5A-B for schematics of training using curriculum learning).[3] showed that machines also learn best in stages this way. Neural networks trained in stages are trained faster, with fewer training iterations. Additionally, neural networks trained this way can also reach more optimal solutions [3, 10, 13]. Since curriculum learning can result in faster training (see Figure 3.5C), some tasks that were previously computationally intractable are now feasible [3].

Because PsychRNN tasks are modular and object oriented, as described in Section 3.4, we were able to design an intuitive framework for curriculum learning that is easy to use even without TensorFlow. Curriculum learning is implemented by passing a curriculum object to the RNN model when training is executed. Although very flexible and customizable, the simplest form of the curriculum object can be instantiated solely with the list of tasks that one wants to train on sequentially (fig 3.S4 which corresponds to the curricula used in Figure 3.5C-D).

Little computational neuroscience research exists to test the importance of curriculum learning. PsychRNN includes an easy-to-use implementation of curriculum learning (fig 3.S4), so researchers can utilize and experiment with curriculum learning with minimal startup cost.

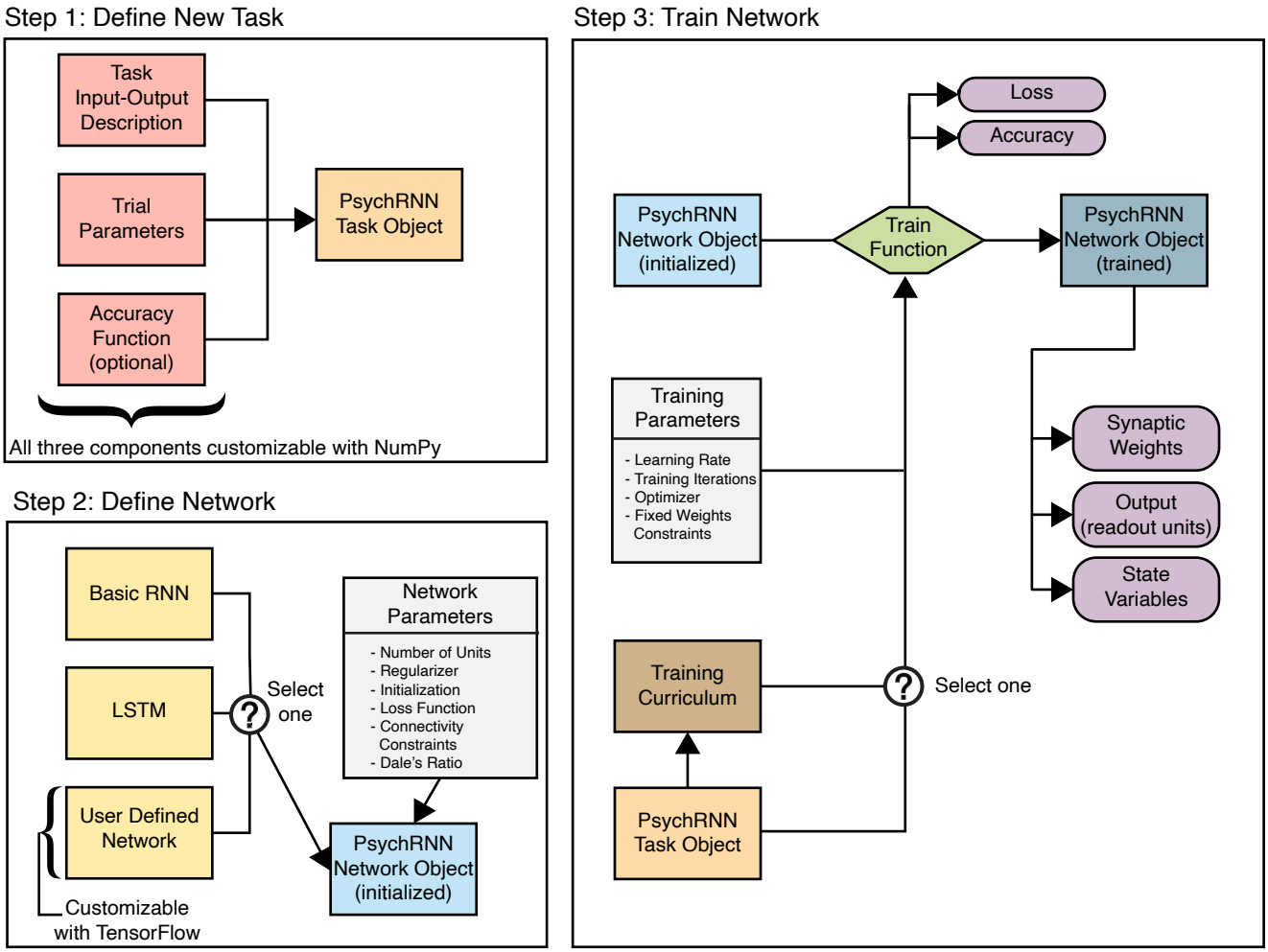


Figure 3.6: **Package Structure and Network Equations** (Step 1) Defining a new task requires defining two Python functions with Numpy. One, `trial_function` describes the task input and output. The other, `generate_trial_params` defines the parameters for a given trial. Optionally, one can also define an accuracy function describing how to calculate if a trial was successful. (Step 2) The Backend. First, the model, or network architecture, is selected. Currently, versions of a normal RNN and LSTM are implemented — more models or architectures can be defined using TensorFlow. Then, that model is instantiated with a dictionary of parameters. That dictionary of parameters must include the number of recurrent units, but it may also include specifications of loss functions, initializations, regularizations, or biological constraints. When the network is trained, training specifications, such as the optimizer or curriculum can be specified. This creates a trained neural network, parameterized as one chooses. If any parameter is not set, a default is used. (Step 3) When a network is trained, measures of performance indicate the speed at which the network is learning at regular epochs (time points or iterations) throughout training. We include two measures of performance: loss and accuracy. Loss is a metric defined by the loss function chosen (by default, loss is the mean squared error over the target output and network output). Loss is a typical metric used in machine learning, and a lower loss value corresponds to better performance.

Figure 3.6: (Previous page.) Optimization of the network weights is performed on the loss. Accuracy is an optional metric that is defined in the task definition. We use accuracy as a more biologically relevant measure of performance. On a given trial, the accuracy value is either one (success) or zero (failure). In contrast, loss on a given trial is a positive real-numbered value. Accuracy is calculated over multiple trials to obtain a ratio of trials correct to trials incorrect. As discussed in Section 3.4, once the network is trained, the synaptic weight matrix can be saved out, and state variables and network output can be generated for any given trial.

Curriculum learning is perhaps especially interesting to our main audience as a way to generate principles hypothesis regarding the efficiency of their own animal training protocols, and how training protocols could potentially reveal task factorizations useful for animal training.

The curriculum class included in PsychRNN is very flexible and allows for extensive customization. By default, accuracy, as defined within a task, is used to measure the performance of the task. When the performance surpasses .9, the network starts training with the next task. However, it is possible that one might want to advance the training stage at a different threshold than .9, or advance the stage at different thresholds depending at which stage the network is at within the curriculum. The curriculum object thus includes an optional input array, `thresholds` for specifying the cutoff for the performance. Additionally, there are many reasonable ways to determine when to advance the curriculum stage other than accuracy, for example using loss, number of iterations, or some other measure. We include an optional `metric` function that can be passed into the curriculum class to define a custom measure of performance.

The implementation of curriculum learning included in PsychRNN is easy to use and customizable. Customizing the Curriculum as described above requires only knowledge of Python and NumPy.

NumPy Simulation

One limitation of specifying RNN networks in the TensorFlow language is that in order to run a network the inputs, outputs, and computation need to take place within the TensorFlow framework. This limits researchers unversed in TensorFlow in their ability to design and implement experiments on their trained PsychRNN models.

To mitigate this, we have included a NumPy simulator which takes in a PsychRNN model object and Task Object and runs the network in NumPy. Using this simulator, researchers can explore how their network may respond to a variety of inputs. Further, of increasing interest in neuroscience, and thereby in computational neuroscience, is the impact of perturbation experiments on neural computation. By providing a NumPy simulator we hope that research groups will be able to modify it such that they can generate and produce virtually any network or state space perturbation they might require for their research question.

Comparison to Other Frameworks

PsychRNN was designed to facilitate the growing use of RNNs as a model for neural computation. As such it provides several advantages over alternative high level frameworks.

Most similar to PsychRNN is the PyCog package ([16]), another python package for training RNNs designed for neuroscientists. In comparison to PyCog, PsychRNN prevents several key advantages. First PyCog was built off of a framework which is no longer supported. Second PyCog has no native implementation of curriculum learning. Third task definitions in PyCog are not themselves modular making the types of experiments trivial in PsychRNN more laborious and cumbersome. Lastly, PyCog utilizes a built in vanilla stochastic gradient descent algorithm, while PsychRNN allows users to select any

Table 3.1: Comparison to Existing Packages and Other Alternatives.

Implemented Features	PsychRNN	PyCog [16]	Keras [5]
Key Advantage	Curriculum learning, modular task and network, supported Backend	Biological Constraints on RNNs	Works with multiple Backends
Language	Python	Python	Python
Backend (currently supported?)	TensorFlow 1 (maintenance mode only) and 2 (yes)	Theano (no)	TensorFlow 1 (maintenance mode only), TensorFlow 2 (yes), Theano (no), CNTK (yes)
Biological Constraints Supported	Dale's Principle, Connectivity Patterns, Fixed Weight Training	Dale's Principle, Connectivity Patterns, Fixed Weight Training	Not supported
Curriculum learning supported?	yes	no	N/A
Modular task definition?	yes	no	N/A
Object Oriented Framework?	RNN, Task	RNN	Network Layers
LSTM	Built in support	Not supported	Built in support
Optimizer	TensorFlow built in options with implemented regularizers	SGD with implemented regularizers	Built in options
GPU support	Yes	Yes	Yes
Supports PyTorch	No	No	No

optimizer available in the TensorFlow package.

Outside of PyCog the other main option for research groups is to use a high level wrapper of TensorFlow, such as Keras, not specifically designed for neuroscientific research. Importantly, these frameworks do not come with any substantial ability to implement biological constraints. Users interested in testing the impact of such constraints would need to modify the native Keras layer objects themselves, a non trivial task. In addition Keras does not provide a frameworks for modular task definition, requiring the user to do the job of translating inputs and outputs into a form compatible with the model.

PsychRNN by close integration with the TensorFlow framework manages to maintaining much of the power and flexibility of traditional machine learning frameworks while also providing custom built utilities specifically intended for the challenges of a RNN neuroscience research program.

Code Availability

The PsychRNN software described in the paper is freely available online at github.com/murraylab/PsychRN

All data and figures included were produced on a MacBook Pro (Retina, 13-inch, Early 2015) with 8 GB of RAM and 2.7 GHz running macOS Catalina 10.15.5 in an Anaconda environment with Python 3.6.9, NumPy 1.17.2, and TensorFlow 1.14.0.

Bibliography

[1] Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Curr Opin Neurobiol*, 46:1–6, 10 2017. 1, 20

[2] Manuel Beiran and Srdjan Ostojic. Contrasting the effects of adaptation and synaptic filtering on the timescales of dynamics in recurrent networks. *PLoS computational biology*, 15(3):e1006893, 2019. 1

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA, 2009. ACM. 21

[4] K H Britten, M N Shadlen, W T Newsome, and J A Movshon. The analysis of visual motion: a comparison of neuronal and psychophysical performance. *J Neurosci*, 12(12):4745–65, Dec 1992. 13

[5] François Chollet et al. Kera. `https://github.com/fchollet/keras`, 2015. 25

[6] J. C. Eccles, P. Fatt, and K. Koketsu. Cholinergic and inhibitory synapses in a pathway from motor-axon collaterals to motoneurones. *The Journal of Physiology*, 126(3):524–562, 1954. 19

[7] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep

feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. 17

[8] Vishwa Goudar and Dean V Buonomano. Encoding sensory and motor patterns as time-invariant trajectories in recurrent neural networks. *Elife*, 7:e31134, 2018. 1

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 17

[10] 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*, 114(13):3521–3526, 2017. 21

[11] Satoshi Kuroki and Takuya Isomura. Task-related synaptic changes localized to small neuronal population in recurrent neural network cortical models. *Frontiers in Computational Neuroscience*, 12:83, 2018. 1

[12] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, Nov 2013. 1

[13] A Pentina, V Sharmanska, and C H Lampert. Curriculum learning of multiple tasks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5492–5500, June 2015. 21

[14] Kanaka Rajan and L. F. Abbott. Eigenvalue spectra of random matrices for neural networks. *Phys. Rev. Lett.*, 97:188104, Nov 2006. 19

[15] R Romo, C D Brody, A Hernández, and L Lemus. Neuronal correlates of parametric working memory in the prefrontal cortex. *Nature*, 399(6735):470–3, Jun 1999. 15

[16] H Francis Song, Guangyu R Yang, and Xiao-Jing Wang. Training excitatory-inhibitory recurrent neural networks for cognitive tasks: A simple and flexible framework. *PLoS Comput Biol*, 12(2):e1004792, Feb 2016. 2, 19, 24, 25

[17] David Sussillo and L.F. Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544 – 557, 2009. 17

[18] David Sussillo, Mark M Churchland, Matthew T Kaufman, and Krishna V Shenoy. A neural network that finds a naturalistic solution for the production of muscle activity. *Nat Neurosci*, 18(7):1025–33, Jul 2015. 1

[19] Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. A simple way to initialize recurrent networks of rectified linear units. 04 2015. 17

[20] Marcel van Gerven. Computational foundations of natural intelligence. *Frontiers in Computational Neuroscience*, 11:112, 2017. 1

[21] Yuan Wang, Yao Wang, and Yvonne W Lui. Generalized recurrent neural network accommodating dynamic causal modeling for functional mri analysis. *NeuroImage*, 178:385–402, 2018. 1

[22] Daniel L. K. Yamins, Ha Hong, Charles F. Cadieu, Ethan A. Solomon, Darren Seibert, and James J. DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23):8619–8624, 2014. 1

[23] Daniel LK Yamins and James J DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356, 2016. 1

3.5 Appendix

Supplementary Figures

```python
class SimplePD(Task):
  def __init__(self, dt, tau, T, N_batch):
    super(SimplePD,self).__init__(2, 2, dt, tau, T, N_batch)
  def generate_trial_params(self, batch, trial):

    # ----------------------------------
    # Define parameters of a trial
    # ----------------------------------
    params = dict()
    params['coherence'] = np.random.exponential(scale=1/5)
    params['direction'] = np.random.choice([0, 1])

    return params

  def trial_function(self, t, params):
    stim_noise = 0.1
    onset = self.T/4.0
    stim_dur = self.T/2.0

    # ----------------------------------
    # Initialize with noise
    # ----------------------------------
    x_t = np.sqrt(2*self.alpha*stim_noise*stim_noise)*np.random.randn(self.N_in)
    y_t = np.zeros(self.N_out)
    mask_t = np.ones(self.N_out)

    # ----------------------------------
    # Retrieve parameters
    # ----------------------------------
    coh = params['coherence']
    direction = params['direction']

    # ----------------------------------
    # Compute values
    # ----------------------------------
    if onset < t < onset + stim_dur:
      x_t[direction] += 1 + coh
      x_t[(direction + 1) % 2] += 1

    if t > onset + stim_dur + 20:
      y_t[direction] = 1.

    if t < onset + stim_dur:
      mask_t = np.zeros(self.N_out)

    return x_t, y_t, mask_t
```

Figure 3.S1: **Example Task Definition** The Code Sample above defines a simple PD task. `generate_trial_params` selects the coherence and direction on a trial by trial basis. `trial_function` sets the input, target output and output mask depending on the time in the trial and the parameters passed in.

```
from psychrnn.tasks.delayed_discrim import DelayedDiscrimination

for i in range(3):
  for j in range(3):
    dd = DelayedDiscrimination(dt = 10, # simulation time step
              tau = 100,  # unit time constant
              T = 2000, # trial length
              N_batch = 1, # number of trials per update
              delay_duration = (j+1) * 250, # delay length
              decision_duration = 250, # decision length
              onset_time = 125, # first stimulus onset time
              stim_duration_1 = (i+1)/3 * 500, # stim 1 length
              stim_duration_2 = (i+1)/3 * 500)  # stim 2 length
    x, target_output, mask, trial_params= dd.get_trial_batch() # get task
```

Figure 3.S2: **Task Modularity: Task Structure** The code sample above produces all of the data shown in Figure 3.3B. Although each plot shown in Figure 3.3 .b has a slightly different task specification, we can iterate through all of them easily using the object-oriented modular task definitions enabled by PsychRNN. This produces clean, easy-to-read and -understand code compared to non-modular alternatives.

```
from psychrnn.tasks.match_to_category import MatchToCategory

for i in range(3):
  for j in range(3):
    mc = MatchToCategory(dt=10, # simulation time step
            tau = 100, # unit time constant
            T = 2000, # trial length
            N_batch = 1, # number of trials per training update
            N_in=(i+1)*4, # number of network inputs
            N_out =(j +1)*2) # number of network outputs
    x, target_output, mask, trial_params = mc.get_trial_batch() # get task
```

Figure 3.S3: **Task Modularity: Inputs & Outputs** The code sample above produces all of the data shown in Figure 3.3D. Although each plot shown in Figure 3.3D has different numbers of inputs and outputs, we can iterate through all of them easily using the object-oriented modular task definitions enabled by PsychRNN. This produces clean, easy-to-read and -understand code compared to non-modular alternatives.

```
1  from psychrnn.tasks.perceptual_discrimination import PerceptualDiscrimination
2  from psychrnn.Backend.curriculum import Curriculum
3
4  coherences = [.7, .5, .3, .1]
5  task_list = [PerceptualDiscrimination(dt=10, # simulation time step
6                tau = 100, # unit time constant
7                T = 2000, # trial length
8                N_batch = 128, # number of trials per update
9                coherence = coh # coherence of trials
10              ) for coh in coherences]
11 curriculum = Curriculum(task_list, # list of tasks that make up the curriculum
12         output_file="./accuracies", # path to save metric value
13         metric_epoch= 5 # interval to check the metric
14          )
15 train_params = {
16   "save_weights_path": "./weights.npz", # path to save trained network weights
17   "training_iters": 100000, # maximum number of training iterations
18   "loss_epoch": 5, # how often to calculate loss
19   "curriculum": curriculum # Curriculum object
20 }
21
22 # curriculum-training specific wrapper of the train function
23 losses , training_time, initialization_time = model.train_curric(
24          train_params # training parameters
25          )
```

Figure 3.S4: **Curriculum Learning Code** The code sample above trains an RNN on a sequence of perceptual discrimination tasks with decreasing coherence. The network starts by learning to perform the task with high coherence. Once the network reaches 90 percent accuracy on a given task, the network initiates training on the next task. This continues until the network has reached 90 percent accuracy on the final task—in this case, the lowest coherence perceptual discrimination task. Curriculum learning is done by making a list of tasks that form the curriculum (line 4-10), and then passing that list in to the Curriculum class to form a curriculum object (line 11-14). That curriculum object is then included in the training parameters dictionary (line 15-20), and when the network is passed those training parameters for training, the network will be trained using the curriculum, or sequence of tasks defined in lines 4-10.

Chapter 4

Geometry of neural computation unifies working memory and planning

Daniel Ehrlich, John Murray

Ehrlich, D. B., Murray, J. D. (2021). Geometry of neural computation unifies working memory and planning. bioRxiv.

Abstract

Real-world tasks require coordination of working memory, decision making, and planning, yet these cognitive functions have disproportionately been studied as independent modular processes in the brain. Here, we propose that contingency representations, defined as mappings for how future behaviors depend on upcoming events, can unify working memory and planning computations. We designed a task capable of disambiguating distinct types of representations. Our experiments revealed that human behavior is consistent with contingency representations, and not with traditional sensory models of working memory. In task-trained recurrent neural networks we investigated possible circuit mechanisms for contingency representations and found that these representations can explain neurophysiological observations from prefrontal cortex during working memory tasks. Finally, we

generated falsifiable predictions for neural data to identify contingency representations in neural data and to dissociate different models of working memory. Our findings characterize a neural representational strategy that can unify working memory, planning, and context-dependent decision making.

4.1 Introduction

In time-varying environments, flexible cognition requires the ability to store and combine information across time to appropriately guide behavior. In commonly used delay task paradigms, a transient sensory stimulus provides information which the agent must maintain internally across a seconds-long mnemonic delay to guide a future response [16, 37, 52]. Working memory is a core cognitive function for the active maintenance and manipulation of task-relevant information for subsequent use. To guide flexible behavior, contents of working memory must interface with other cognitive functions such as planning and context-dependent decision making. Yet within neuroscience, these functions have been studied largely independently, and it remains poorly understood how the brain coordinates these processes in the service of goal-directed behavior.

Internal representations related to cognitive states can be revealed through recording neural activity during delay tasks in animals and humans. Neurons in the prefrontal cortex exhibit content-selective activity patterns during mnemonic delays of working memory tasks[20, 12]. Selective delay activity during working memory is most commonly interpreted as representing features of sensory stimuli which can be processed to guide a later response. In the dominant conceptual framework, the proposed cognitive strategy thereby uses working memory representations that are fundamentally sensory in nature[20, 12]. The sensory strategy for working memory has been challenged by observations of prefrontal delay activity that better correlate with diverse task variables, including actions,

expected stimuli, and rules[37, 38, 52]. Furthermore, a growing literature characterizes substantial nonlinear mixed selectivity of features in prefrontal cortex, which can in principle support context-dependent behavior[41, 26, 7]. These diverse observations suggests a more unified framework is needed to account for the computational roles of working memory, planning, and context-dependent decision making.

Computational modeling has been fruitfully applied to examine potential neural circuit mechanisms supporting cognitive functions, including working memory and decision making[54]. Working memory functions are commonly modeled with distinct modular circuits, which do not account for mixed selectivity in prefrontal cortex and assume that working memory maintains sensory representations[8, 54]. A complementary modeling approach utilizes artificial neural network models which are trained to performed cognitive tasks[24]. In contrast to hand-designed models, task-trained recurrent neural network (RNN) models can perform delay tasks without requiring assumptions about structured circuit architectures or the form of working memory representations[3]. This approach is therefore well suited to examine computational mechanisms through which working memory representations can support flexible computations[40, 30, 56, 13].

In this study, we investigate the implications of a cognitive strategy in which internal states represent plans, rather than perceptions or actions. Specifically, our theoretical framework defines contingency states based on how future behaviors depend on upcoming events. We designed a task paradigm to dissociate contingency-based strategies from sensory- or action-based strategies for working memory and found that contingency strategy better explained human behavior in this task. RNN models trained on the task develop contingency-based solutions, and their internal representations capture diverse phenomena of neural activity during working memory. Lastly, we provide falsifiable predictions for neural activity to distinguish contingency representations from alternative computational schemes. Taken together, our study presents a theoretical framework for how working

memory supports planning for temporally extended cognition and flexible behavior, which explains disparate behavioral and neural observations and is experimentally testable.

4.2 Results

Conditional delayed logic task

In many commonly used delay tasks, there are correlations among sensory stimuli, responses, computational demands or rules which prevent dissociable attribution of neural activity to specific cognitive task variables[16, 31]. We therefore sought a task that involves (i) computation on informational inputs separated by a delay, (ii) exact intermediate computational states which re-occur frequently, (iii) trials for which the correct response can be predicted during the delay on some trials and not on others, and (iv) decorrelation of action, sensory, rule and computational state information.

To meet these demands we designed the conditional delayed logic (CDL) task, which applies a binary classification to two binary stimuli separated by a delay (**Fig. 4.1a**). We refer to the pre-delay and post-delay stimuli as ”Cue A” and ”Cue B”, respectively. The rules can be defined as Boolean logical operations, and one rule is presented tonically throughout each trial’s duration. On each trial, the agent is presented with a task rule, Cue A and Cue B and must generate an associated response. For example, in the OR rule the agent’s response should be ”1” if either cue is equal to ”1”. From 16 possible two-bit binary classifications, we will focus on 10 rules (**Fig. 4.1b**) for which the response is dependent on cues but independent of cue order. Rule, Cue A and Cue B are varied randomly across trials.

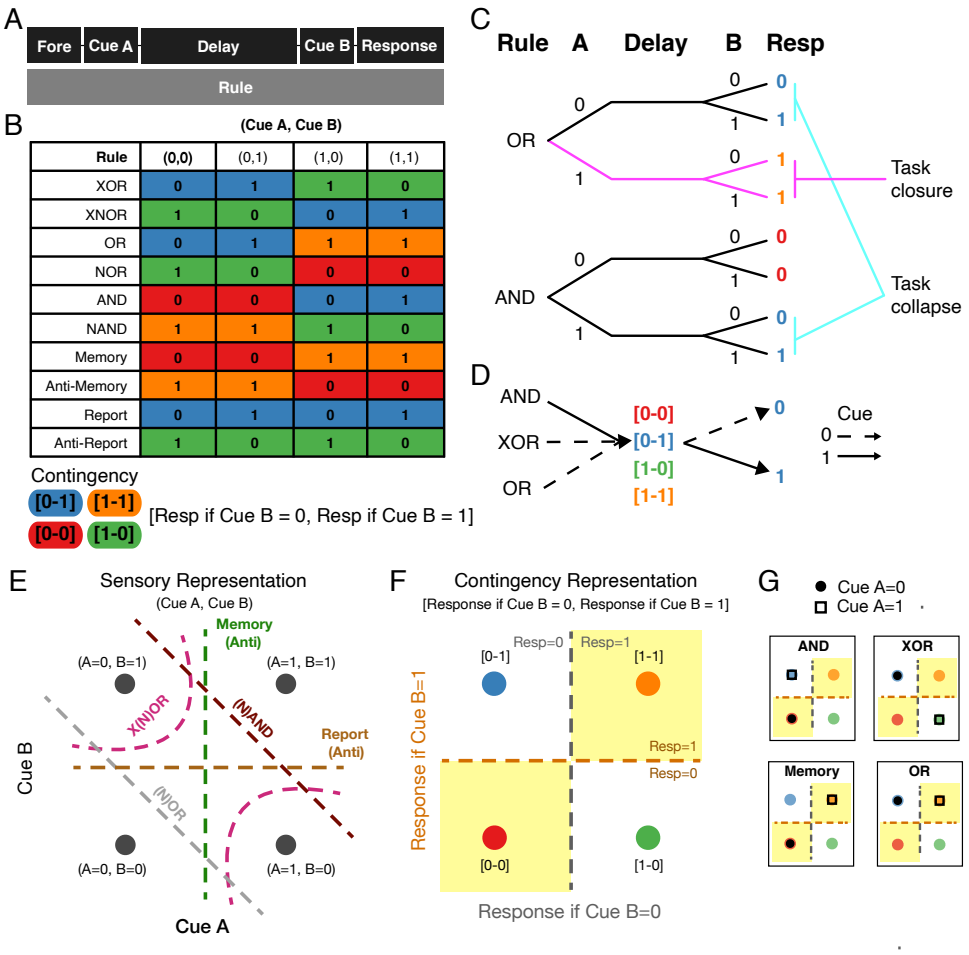


Figure 4.1: **Conditional Delayed Logic (CDL) task and contingency states.** **(a)** Time course of events in the CDL task. **(b)** The contingency table of responses by rule and (Cue A,Cue B) pair. Color indicates the contingency of that condition. **(c)** A schematic example of task closure and collapse between tasks and conditions. Top, tree structure of an OR trial, Bottom, tree structure of AND trial. The magenta lines indicates a condition (OR, Cue A=1) in which the trial is termed "closed" due to the fact that the correct response is independent of Cue B. The cyan lines indicate two conditions (OR, Cue A=0; AND, Cue A=1) which "collapse" during the delay. This is because after Cue A, their response patterns to Cue B becomes identical: in both cases if B=0 the correct response is 0, and if B=1 the correct response is 1. Color of responses indicate associated contingency. **(d)** Example solutions of the CDL task through contingency states. Solid lines represent a Cue of 0, and dashed lines represent a Cue of 1. All three tasks (AND, Cue A=0; XOR, Cue A=1; OR, Cue A=1) have a [0-1] contingency and therefore can be solved together after the delay.

Contingency representations

To perform the CDL task, the agent must maintain a representation of task information across the delay. One might expect the agent to maintain the stimulus identity of Cue A,

which would be sufficient to solve all trials. In this 'sensory strategy', which is commonly assumed in models of working memory, the task-relevant information is maintain across the delay to guide a response in conjunction with information in Cue B and rule[20, 53]. An alternative 'action strategy' is possible for trial conditions in which the response can be preplanned during the delay[17, 37]. Furthermore, the agent may represent the rule identity which is presented during the delay[52]. These representations are not exclusive.

In contrast to sensory, action or rule representations, an alternative strategy for the CDL task, and other working memory tasks, uses what we call 'contingency representations'. Contingencies are defined by mappings from the upcoming cues to responses (**Fig. 4.1b**). For example, if the rule is OR and Cue A is "0", then the contingency state during the delay is the mapping from Cue B being "0" or "1" to the correct response being "0" or "1", respectively. Throughout this paper we will refer to contingency using the notation [R_0-R_1] where R_0 and R_1 are the response targets for Cue B being "0" and "1", respectively. We would therefore say the above trial has a [0-1] contingency state, which could be represented in working memory.

The CDL task provides two important features that can be utilized by contingency representations: task 'closure', which gives insight into conditional computational demands; and task 'collapse', which enables identification of specific computationally relevant states. Closure describes the extent to which all information necessary to perform the next step of the task is already presented. In the CDL task, closure is defined as a condition in which the correct response can be decided from the rule and Cue A alone, and therefore the response can be pre-planned during the delay before Cue B. For example, if the rule is OR and Cue A is "1", then regardless of Cue B the correct response will always be "1" (**Fig. 4.1c**). This enables the agent to plan the response directly following Cue A.

Collapse describes the condition in which the agent reaches the same computational branch point in a multi-stage problem independent of the prior path. In the CDL task,

collapse describes when two different rule and Cue A conditions have the same contingency state with regard to Cue B. One example of this is in the OR and AND tasks: The [0-1] contingency state is reached both by the OR rule with Cue A of ”0”, and by the AND rule with Cue A of ”1” (**Fig 4.1c**). In this way two trials that share neither rule nor Cue A can share a contingency state, demonstrating functional collapse. As described later, this property will be crucial in dissociating contingency representations from sensory representations in neural delay activity.

Computing through contingency representations

We found that one advantage of the contingency representation is that it reduces the overall classifier complexity problem of the task. The sensory representation requires 12 classifier hyperplanes to implement the ten rules, while the contingency representation requires only two (**Fig. 4.1e,f**). The contingency representation acts as a modular solution to the CDL task in which dynamics and representations are re-used in order to solve multiple conditions (**Fig. 4.1g**). Contingency representations solve the CDL task in two steps, a first step mapping from rule and Cue A to a contingency state and a second step mapping from contingency and Cue B to response. This decomposition turns a single complex classification problem into two simple ones. Interestingly, this complexity reduction can be formalized using a straightforward linear matrix decomposition between pre-delay features (rule and Cue A) and post delay features (Cue B and target) (**Supplementary Fig. 4.S1**), which results in the contingency basis as described above (**Fig. 4.1f**).

Human behavior matches contingency strategy

While there are theoretical advantages to computing with contingency representations compared to sensory representations, it is unclear whether humans would use contingency

as a strategy to solve the CDL task. To identify whether contingency is present in human behavior, we investigated the impact that closure and collapse had on response times (RTs) in a five-rule variant of the CDL task (**Fig. 4.2a**). We hypothesized that 'closed' trial conditions with task closure would elicit shorter RTs than 'open' conditions without closure, because on closed trials participants can pre-plan their motor response during the delay, whereas on open trials they must wait for Cue B to form their response.

We tested 17 human participants on the CDL task, measuring accuracies and RTs. Participants learned to perform the task with a high mean accuracy ($97\pm1\%$) across all rules (**Fig 4.S2**). To investigate cognitive and computational strategies underlying performance of the CDL task, we measured and compared RTs across task conditions both between rules and between Cue As (**Fig. 4.2b**). We found that RTs varied strongly across trial conditions and broadly sort into two groups according to closed vs. open conditions, with closed trials having a significantly shorter mean RT than open trials ($t = -12.04$, $p = 10^{-7}$).

Crucially, the OR and AND subtasks, which vary on their closure status on a trial-by-trial basis, indicate that this RT difference between closed and open trials is not due to rule or cue differences. Within OR and AND, there remained a significant difference in RT between open and closed trials (OR; $t = -6.4$, $p = 5 \times 10^{-5}$, AND; $t = -7.1$, $p = 2 \times 10^{-5}$) (**Fig. 4.2b**). Within-individual analysis found that only for OR and AND was there a significant effect of Cue A on RT for the majority of participants (**Fig. 4.2c**), which would be expected if closure was driving RT. We observed that the closure effect, the impact of closure on RT, grew over the course of the session (OR: $p = 0.018$, AND: $p = 0.009$) (**Figs. 4.2d, 4.S2d**), which suggests that the effect of closure can be learned through experience and be enhanced through training. The difference in RT between closed vs. open trials implies that they were capable of flexible updating of plans within a trial as they processed information from the rule and Cue A. This conditionally variant RT pattern

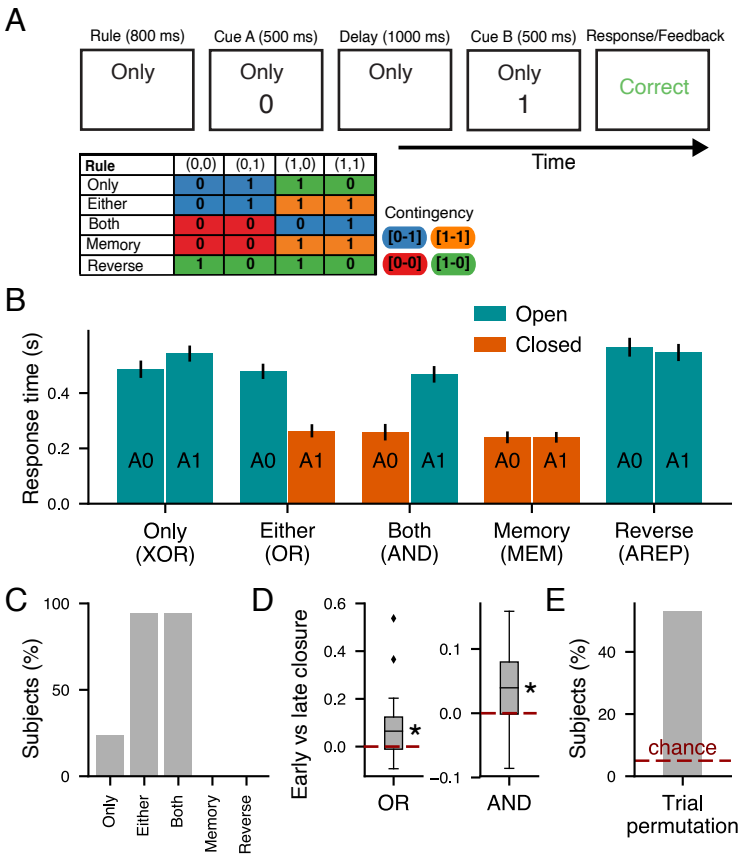


Figure 4.2: **Human behavior on CDL task supports contingency-based strategies. (A)** Top: The procession of stimuli shown to participants during an example trial (Only (XOR), (Cue A=0, Cue B=1)). Bottom: The table of responses by rule and (Cue A,Cue B) pair. Color indicates the contingency of that condition. **(B)** Mean response time by Cue A and rule. Error bars represent standard error of the mean across participants. Magenta bars are open trial conditions and orange bars are closed trial conditions. Logical rule names in parenthesis below labels. **(C)** Percent of participants with a significant ($p < 0.05$) difference in mean RT between Cue A=0 and Cue A=1 within a given rule. **(D)** The difference in closure effect, defined as the difference between the mean RT of the open condition and closed condition, between early trials (first two blocks of the task) and late trials (the last two blocks). Plotted separately for the OR and AND rules. **(E)** The percent of participants for whom contingency explained more RT variance than would be expected by chance for the linear model permutations. Dashed lines represent chance level, and stars indicate $p < 0.05$ (one-sided t-test).

is opposed to sensory strategies for which subtasks of similar complexity ought to take similar times to complete independent of how Cue A interacts with the rule (**Fig. 4.1e**).

While the analysis above indicates evidence against a sensory strategy for working

memory, it does not specifically address whether participants use contingency representations with collapse across task conditions. To investigate the extent to which contingency is an explanatory task variable for human behavior, we used a linear modeling approach to measure the proportions of variance in RTs explainable by contingency, while controlling for effects of closure, response target, rule, and congruency between Cue B and response (see **Methods**). We utilized a permutation method to measure individual-level significance (**Fig. 4.2e**). We found that 53% of our participants (9 of 17) had significant explainable RT variance attributable to contingency, and that such a proportion was substantially higher than would be expected by chance ($p = 3 \times 10^{-8}$) (**Fig. 4.2f**). Collectively, our behavioral findings provide support that in the CDL task humans use contingency-based strategies and not sensory working memory strategies.

Task-trained RNNs utilize contingency representations

To explore how contingency-based computations may be realized in a distributed recurrent circuit, we trained recurrent neural network (RNN) models to perform the full CDL task (**Fig. 4.3a**). Our trained RNN model reached high accuracy across all subtasks (99.3% average accuracy, with all >95%). In order to identify the structure of contingency information in our trained RNNs, we used a simple subspace identification procedure [28]. Using the network state vector from the late-delay epoch (just before Cue B onset) we identified two contingency-coding axes maximally capturing variance in neural states across units in the RNN explained by the target response conditioned on each Cue B stimulus (see **Methods**). We then projected the late-delay state from each trial into this two-dimensional contingency subspace. This representation was found to cleanly and linearly separate between the four possible contingency conditions (**Fig. 4.3b**).

To test the importance of the contingency subspace, we measured the amount of trial-wise variance in late delay state vector captured by the contingency subspace and found

greater than five times as much variance compared to random two-dimensional subspaces (**Fig. 4.3c,d**). We tested the functional relevance of the subspace for task performance through a perturbation approach. Just prior to Cue B onset, we shifted the state of the network by a given magnitude in a random direction which was either within the contingency subspace or orthogonal to the subspace. We found that RNNs suffered greater performance deficits from perturbations within the contingency plane, indicating the functional relevance of the subspace for the CDL task (**Fig. 4.3e**).

One advantage of the contingency representation is that it organizes delay representations such that they are already separable by mappings from Cue B to response. A single selection vector can therefore separate all states into the correct response (**Fig. 4.3f**). We analyzed the average displacement for trials in each of the contingencies during Cue B and found that they closely correspond to a theoretically optimal selection vector (**Fig. 4.3g**). The state-space dynamics following Cue B offset follows a nonlinear trajectory consolidating trials to the appropriate response state (**Fig. 4.3h**), revealing that while the decision problem itself is linearized by contingency representations, task computation in the RNN relies on nonlinear dynamics.

Control analyses training artificial network models on CDL task variants investigated the computational properties leading to contingency representations. First, we trained an RNN on a task variant in which the rule is not presented until the Cue B epoch (**Supplementary Fig. 4.S4**). This RNN perform this late-rule task with high accuracy, but it did not develop any contingency representations during the delay. This demonstrates that RNNs can solve the CDL task using a sensory strategy and that contingency representations are not an artifact of our analysis method. Second, we trained a three-layer feedforward network, in contrast to an RNN, providing task inputs into different layers of the network (**Supplementary Fig. 4.S5**). Contingency coding only emerged as a dominant middle-layer representation when Cue A and rule inputs were provided upstream and Cue B downstream, which is the

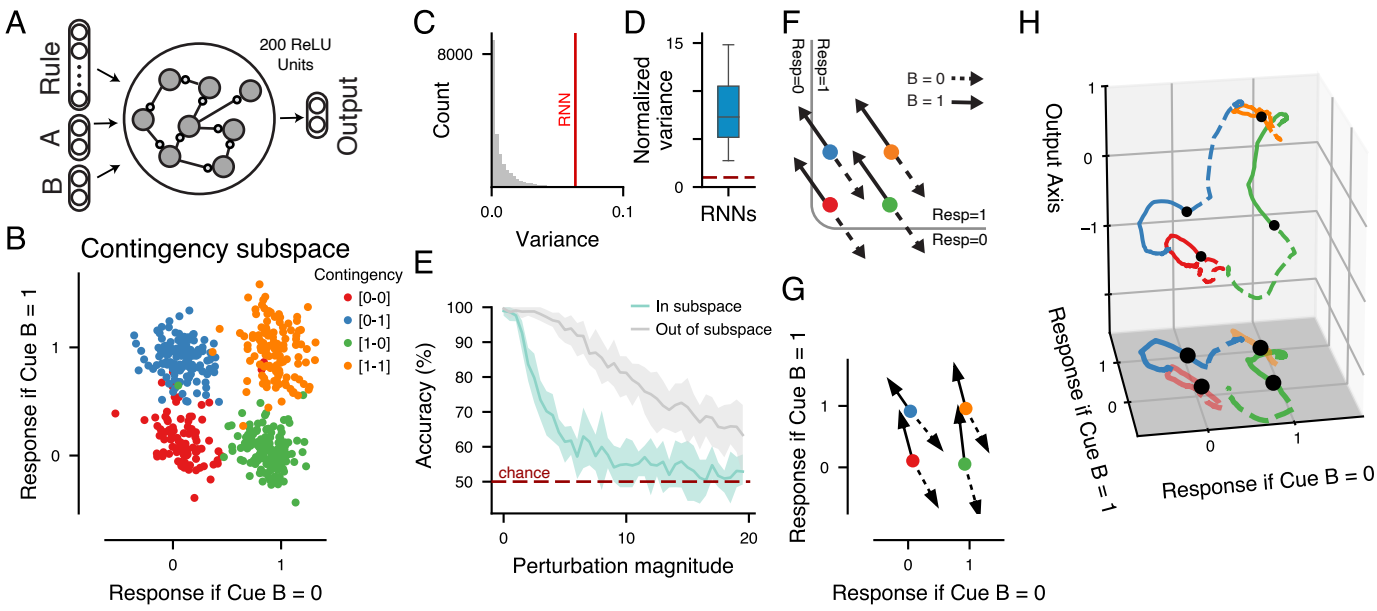


Figure 4.3: **Functional contingency subspace in CDL-trained RNNs. (A)** Schematic of RNN model structure. **(B)** We defined a contingency subspace through a linear regression of unit states during the late delay epoch to define two axes along which 'Response to B=0' and 'Response to B=1' were projected. The contingency subspace was identified on a train set of trials and we used held-out test trials for plots and analysis, for an example RNN. **(C)** A permutation test comparing the amount of variance in the contingency subspace compared to random alternative subspaces. **(D)** Analysis in B is replicated for 20 RNNs over random initializations and plotted is the distribution of variance captured by the contingency subspace compared to the mean variance of random subspaces of equal dimension. **(E)** Mean CDL performance (accuracy) across replicates for a perturbation analysis in which the RNN state was perturbed at the time point prior to Cue B onset by a random vector of a given norm either within the contingency subspace (cyan line) or orthogonal to the subspace (grey line). The perturbation magnitude, norm of the perturbation, was sequentially increased. Shaded regions represent the standard error of the mean across replicate RNNs. Red dashed line represents chance behavior. **(F)** Theoretical schematic of CDL response selection utilizing a contingency representation. Solid arrows represent trajectories caused by Cue B=0 and dashed arrows represent trajectories caused by Cue B=1. The grey separatrix boundary divides regions of state space in which the network will relax to a response of 0 and 1; **(G)** Mean contingency subspace trajectories of the example RNN model from onset to offset of Cue B. Trajectories divided by contingency and Cue B. **(H)** Three-dimensional mean RNN state-space trajectories from Cue B onset to trial end. X- and Y-axes represent the contingency subspace while the Z-axis is the output axis of the example RNN model (i.e., the difference between the magnitude of the two output units).

feedforward analogue of the temporal structure in the CDL task. These analyses suggest that computational stages of information processing, not specific choice of model architecture, robustly determine whether a contingency representation is formed by task-trained network training.

Model neural activity captures neurophysiological findings

While at the population level our trained RNNs can be structurally and functionally identified to be utilizing a contingency-based solution, it is unclear how that is instantiated at the level of individual units and therefore how it might relate to the findings in single-neuron recordings from animals performing working memory tasks. We profiled the selectivity properties of our unit activity vector (**Fig. 4.4a**). While the Cue A averaged traces showed substantial tuning within a rule, we found that between rules the apparent Cue A tuning would invert. For many units, however, this inversion across rules could be substantially explained by contingency. Using a linear model we identified the percent of variance explained in unit state by Cue A, rule and contingency. While we found that most units predominantly encoded Cue A during the stimulus epoch, by the late delay this pattern had changed to primarily encode contingency (**Fig. 4.4b**). Nonlinear dimensionality reduction via UMAP recapitulated the transition from sensory to contingency representations, and showed that contingency and sensory tuning are dominant factors driving unit states in the late vs. early delay epoch, respectively (**Supplementary Fig. 4.S3**).

This transition in information coding during the delay matches prior analyses of single-neuron recordings from prefrontal cortex in monkeys performing working memory tasks. Rainer and colleagues analyzed neurons from alternating match-to-sample and paired-associate working memory tasks, and tested whether prefrontal neurons could better be understood as retrospectively tuned, wherein neuronal activity is tuned to sensory stimuli, or prospectively, wherein activity is tuned to expected future stimuli[38]. They found

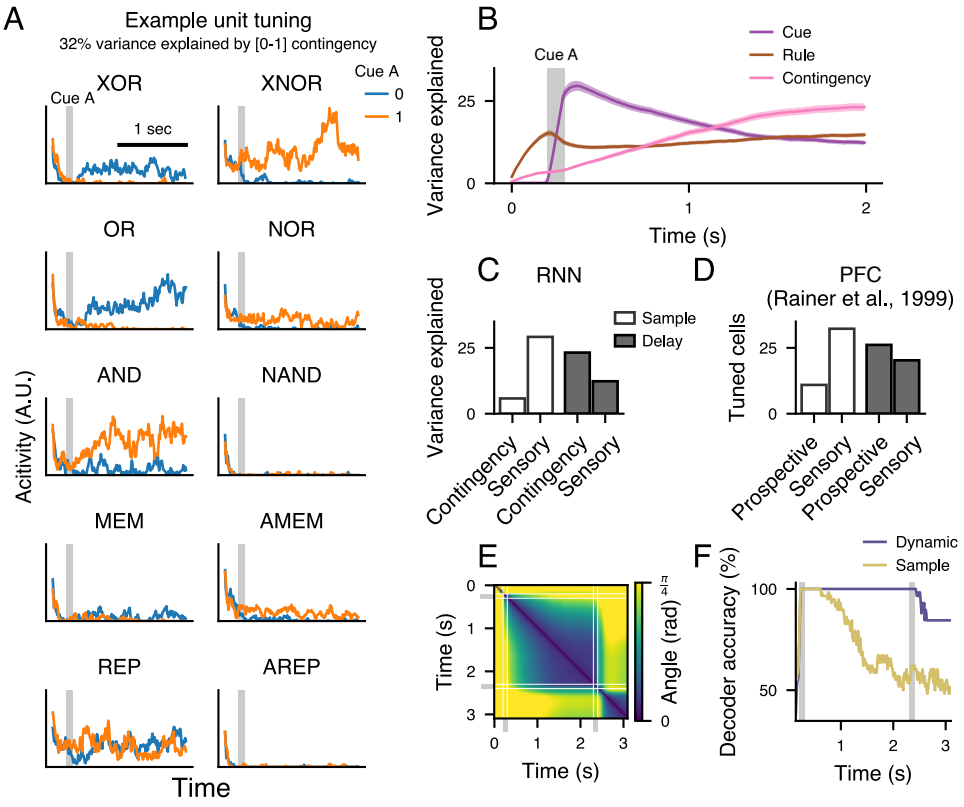


Figure 4.4: **Contingency tuning captures neurophysiological responses in prefrontal cortex. (a)** An example RNN unit with traces representing mean activity in trials divided by Cue A identity and rule. 32% of unit activity variance was explained by a binary predictor of whether it was a [0,1] contingency trial. Shaded grey region indicates the time of Cue A presentation. Traces plotted from trial onset through delay. **(b)** Mean percent of unit variance explained by rule, Cue A and contingency over time. Solid lines represent mean and shaded regions represent the standard error of the mean across replicate RNNs. **(c)** Fraction variance explained by prospective and sensory information for the sample (time point prior to Cue A offset) and delay (time point prior to Cue B onset) epochs in the example RNN model averaged across units. **(d)** Number of recorded neurons from monkey dorsolateral prefrontal cortex (PFC) significantly tuned to prospective and sensory information in the sample and delay epochs[38]. **(e)** Analysis of tuning dynamics across time. Activity in the example RNN was averaged within each condition and then PCA was used to identify the dominant axis of condition tuning. The angle between these axes were measured across each pair of time points. Gray shaded indicate Cue A and Cue B epochs. **(f)** A linear decoder was trained to read out Cue A identity from sample epoch RNN activity and tested throughout the delay epoch. This was compared to a ”dynamic” decoder trained and tested on data from the same time point. Plotted is cross-validated accuracy for each decoder.

a transition from predominantly sensory coding during the cue epoch to predominantly prospective coding during the delay epoch. Our RNN units exhibit a matching transition from retrospective representation, tuned to Cue A, in the cue epoch to a fundamentally prospective representation, tuned to contingency, during the delay epoch (**Fig. 4.4c,d**).

This type of dynamic activity in persistent populations has drawn considerable interest in recent years, with many studies identifying a dynamic decoding axis of perceptual information from sample to delay epochs[44, 49, 9]. The contingency representation provides one possible explanation for this phenomenon. As during the delay the network shifts from a purely sensory to contingency representation if one were to only examine a single task it would appear that the axis along which Cue A is encoded has shifted. We used two analyses to detect dynamic working memory activity. In the first we evaluate change in the principal tuning axis over time using principal component analysis[9] and in the second we train static and dynamic decoders showing a shift in the separatrix necessary to correctly classify the response from trial state vectors (**Fig. 4.4e,f**). In both cases our network matches phenomena previously interpreted as dynamic memory.

Contingency subspace explains heterogenous neuronal tuning results

One question that arises from the above results is how contingency representations would be reflected in typical analyses which measure tuning of neural activity for task variables such as stimulus or rule identity. In the CDL task, tuning for Cue A or rule can be observed in contingency representations when the subtasks generate correlations between cue or rule information and contingency states. These correlations can exist even when Cue A, Cue B and responses are all independent. We examined this in CDL variants with only two rules. For example, the rule pair of Memory and XOR are such that contingency representations yield neural tuning to Cue A but not to rule, due to averaging over contingency states for those subtasks (**Fig. 4.5a**).

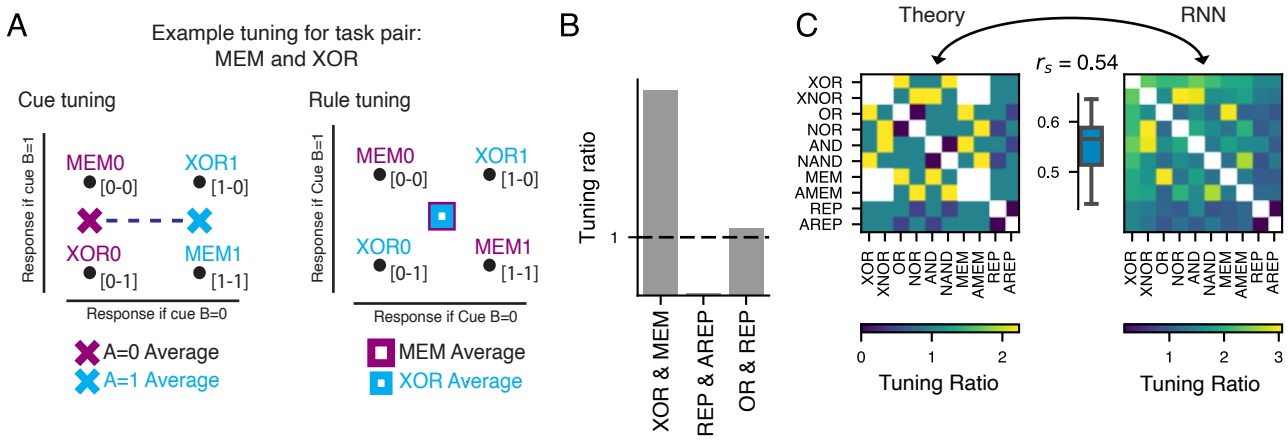


Figure 4.5: **Contingency explains interactions of sensory and rule tuning for pairs of tasks. (a)** Schematic of cue and rule tuning for an example pair of rules (MEM and XOR) utilizing a contingency representation. Task names are followed by a "0" or "1" indicating the Cue A on that trial and are placed in the appropriate contingency for that rule and Cue A pair. Left: X marks denote Cue A averages, with purple for Cue A=0 and blue for Cue A=1. Right: Square marks denote rule averages, with purple for MEM and blue for XOR. The dotted line in cue tuning indicates the difference between average cue states and therefor cue tuning. In contrast the overlap of rule averages indicates this task pair will show little or no rule tuning. **(b)** Cue-rule tuning ratio for three example pairs of rules. Bars show the measured tuning in the RNN when just that rule pair was analyzed. Dashed line represents equal cue and rule tuning. **(c)** Left: Contingency based predictions for cue-rule tuning ratio for each pair of rules in the CDL family defined as the Euclidean distance between cue averages and rule averages. A value greater than 1 indicates more cue tuning, A value less than 1 indicates more rule tuning and a value of 1 represents equal tuning. Right: Tuning ratio between Euclidean distances of cue and rule averages of state space representations of RNNs when analyzed for each pair of rules. Distances averaged across replicate RNNs. Inset: The Spearman correlation of (off-diagonal) predicted and measured tunings ($r_s = 0.54$).

The nonlinear mapping from Cue A and rule to contingency can cause the same unit to be identified as rule- or cue-tuned depending on precisely which task the agent is performing (**Fig. 4.5a,b**). To demonstrate this we analyzed our network units separately for each possible rule pair in the CDL task, forming 100 possible pairs of rules. For each task pair, we could use the expected contingency states to determine theoretical prediction for cue and rule tuning (**Supplementary Fig. 4.S6a,b**). We found that the theoretical contingency-predicted cue:rule tuning ratio significantly correlated with the ratio of tuning

measured from the full CDL-trained RNN model in the late delay epoch (**Fig. 4.5c**).

Together these results show how linear analyses for neural tuning of contingency-coding units can yield apparent tuning to cues and rules. While in any real system, this simplified model of only contingency tuning will not fully explain all sources of variance, the extent to which even a simple model captures between condition tuning variance in our trained high-dimensional and nonlinear RNN demonstrates the robustness of the predictions made by contingency representations. Further, since tuning predictions can be made with small subsets of the CDL task it opens the door to experiments in animals for which it may not be feasible to train on a larger set of rules.

Population coding reflects contingency in model network

To gain insight into how our theory of contingency representations differs from prior theories, and how they may be tested experimentally, we compared against two alternative models. The first is a pure sensory encoding model, in which states uniquely identify different sensory cues presented during the first stimulus epoch and the tonic rule input. The second is a randomly connected network (RCN) model, which generates high-dimensional linearly separable representations of rule and Cue A in order to do arbitrary classification problems[41, 4]. These alternative model classes represent two ends of a continuum from most input structured representation (Inputs) to least structured (RCN). Further, they act as important controls because both models have been used to represent prefrontal computations in the literature [55, 41, 26].

Due to our focus on between-condition representational structure, one useful analysis is representational similarity analysis (RSA)[25]. RSA enables the abstraction of high dimensional data into a set of comparisons between conditions called a representational similarity matrix (RSM). We use Cue A, rule and contingency as theoretical inter-condition templates, e.g. in the Cue A template all conditions with the same Cue A should be similar

(**Fig. 4.6c**). The correlation between RSMs and theoretical templates measures the extent to which activity is structured by those features [25, 22].

The Inputs model by definition only contains input along the Cue A and rule dimensions, but we calculated RSMs for the task-trained RNN and the RCN model (**Fig 4.6a,b**). We then compared these observed similarity matrices to our candidate expected similarity structures constructed by Cue A, rule, and contingency (**Fig. 4.6c**). For the RCN model, Cue A predicts the most between-condition similarity, with rule explaining a lesser fraction and a small negative correlation with contingency. In contrast, the RNN model has inter-condition correlation best explained by contingency (**Fig. 4.6d**). These model predictions for population-level analyses were recapitulated in single-unit analyses by examining the amount of late-delay activity variance captured by a linear model including Cue A, rule, and contingency regressors (**Fig. 4.6e**).

To generate single-neuron predictions we fit a linear model for Cue A, rule and contingency to each unit's activity in the late memory epoch to determine what fraction of variance in activity each regressor explains. The sensory model, by definition has single unit tuning towards Cue A and rule. The RCN model has a majority of units tuned to rule followed by Cue A with few units having substantial variance explained by contingency. The RNN model has the most variance explained by contingency followed by rule and Cue A (**Fig. 4.6e**).

The dimensionality of model state representations provides a complementary view into their geometric structure[1, 15, 2]. We found that the three models make starkly different predictions regarding the dimensionality of the delay representations. The sensory model has dimensionality governed by the inputs directly, and the RCN substantially expands that dimensionality. In contrast, the RNN contracts representations into a lower effective dimensionality which is only slightly higher than the theoretical minimal number of dimensions required by the task (**Fig. 4.6f**).

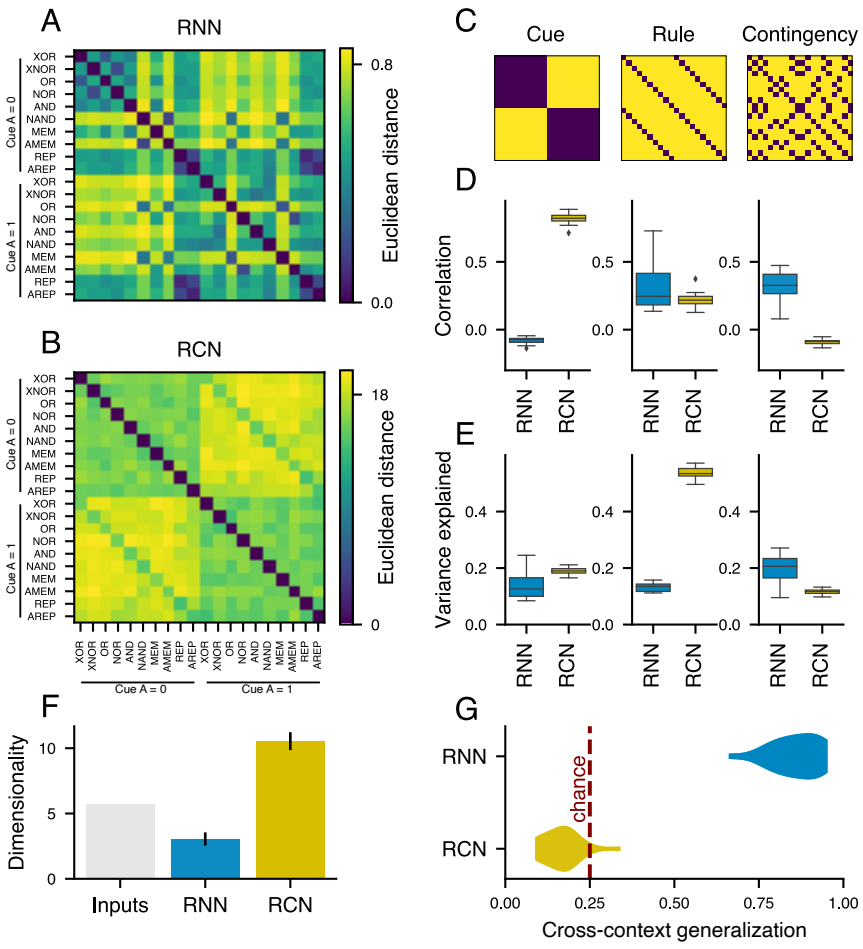


Figure 4.6: **Model comparison and predictions.** **(a, b)** Representational similarity matrices (RSMs) for **(a)** our example RNN model and for **(b)** a randomly connected network (RCN) model capable of performing the CDL task. Model similarity was calculated as the Euclidean distance between averaged late-delay epoch unit activity across conditions for the RNN model and mixed-layer activity for the RCN model. **(c)** Candidate RSMs for representational schemas organized by Cue A, rule and contingency. **(d)** Spearman correlation between the lower triangular portions of the candidate matrices and the RNN and RCN RSMs for each of our 20 replicate networks. **(e)** Mean fraction of variance explained by Cue A, rule and contingency across unit states during the late delay epoch, for the RNN and RCN. Plotted are the distribution of averages across replicate networks. **(f)** Dimensionality of late-delay activity of the RNN model, the RCN model and a "pure" model in which the network represents the Cue A and rule information orthogonally. Error bars represent s.e.m. across replicate networks. **(G)** Distribution of cross-context generalization (CCG) measured across replicate network for our RNN and RCN models. CCG was defined using a contingency subspace classifier. The contingency subspace (**Fig. 4.3b**) was fit as described above with one task condition held out. Then that condition was projected into the subspace and trials were classified based on the quadrant they fell in. Red dashed line represents chance classification.

While the RCN model is designed for information to be generically decodable for a diversity of possible tasks, its high-dimensional representations are relatively unstructured and therefore will not generalize across conditions. We utilized a cross-conditional generalization (CCG) analysis[6] to measure the extent to which the geometry across rule by Cue A pairs was preserved in both our RNN and RCN models. We first fit a contingency subspace, with all trials of a given task condition (rule $\times$ Cue A pair) held out. We then projected the held-out trials into the subspace, and defined a correct classification as a trial falling within the correct subspace quadrant. We found that our RNN models had a mean CCG of 86%, substantially higher than chance ($p = 7 \times 10^{-19}$, one-sided t-test), whereas our RCN models had a mean CCG score of 17% which does not exceed the chance level of 25% (**Fig. 4.6g**).

Testing contingency coding in neural data

The analyses above are sufficient to characterize contingency encoding data generated for a known model class. We next examined how to statistically test for the presence of contingency coding in neural data in which we cannot know the process by which representations are generated. To avoid distributional assumptions about a neural dataset, we devised a non-parametric partitioned variance approach to test whether a given dataset has more contingency-based structure than would be expected based on the amount of other similar nonlinear structure captured in that data (**Fig. 4.7a**).

The analyses above are sufficient to characterize contingency encoding data generated for a known model class. We next examined how to statistically test for the presence of contingency coding in neural data in which we cannot know the process by which representations are generated, e.g. in experimental datasets of neural recordings. Contingency coding generated by a specific form of nonlinear interaction between task variables (e.g., rule and cue). To avoid distributional assumptions about a neural dataset, we devised a

non-parametric partitioned variance approach to test whether a given dataset has more contingency-based nonlinear structure than would be expected based on the amount of other nonlinear structure in the data (**Fig. 4.7a**).

By shuffling contingency mappings across conditions (here, rule-cue pairs) into "psuedo-contingencies" and testing their ability to explain mean unit state variance, we can build a null hypothesis for how well we would expect a nonlinear interaction similar in structure, but different in specific conditions, to contingency would explain the data. Comparing random permutations to the measured quantity from the actual contingency mappings test whether the dataset is better explained by contingency than would be expected by chance independent of the general nonlinearity, feature mixing, or noise structure in the representation. We validated this approach with our RNN and RCN models, finding that our RNN model was better explained by contingency than chance ($p<0.001$) while our RCN model was not (**Fig. 4.7b,c**). Each replicate RNN model exhibited at least twice as much mean unit variance explained by contingency than the mean of its pseudo-contingency null distribution (**Fig. 4.7d**).

By shuffling contingency mappings into "psuedo contingencies" and testing their ability to explain mean unit state variance we can build a null hypothesis for how well we would expect a non-linear interaction similar in structure but different in specific conditions to contingency would explain the data. By comparing our random permutations to the measured quantity from our actual contingency mappings we can then test whether our data is better explained by contingency than would be expected by chance independent of the general non-linearity, feature mixing or noise structure in the representation. Validating this approach, we found that our RNN model was better explained by contingency than chance ($p<0.001$) while our implementation of the RCN model was not (**Fig. 4.7b,c**). Each of our RNN models demonstrated at least twice as much mean unit variance explained by contingency than by the mean pseudo-contingency set (**Fig. 4.7d**).

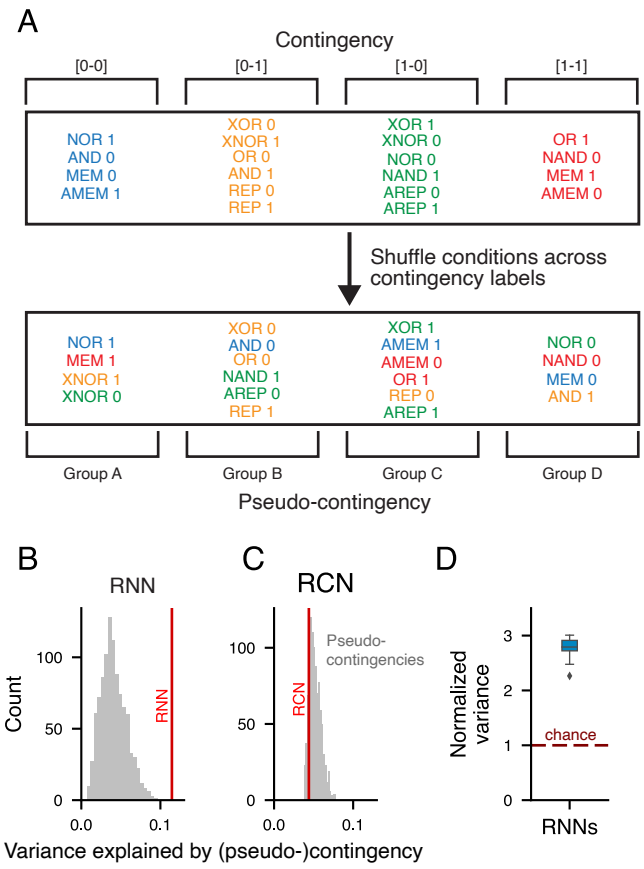


Figure 4.7: **Partitioned variance analysis to test contingency coding.** **(a)** Schematic of structure preserving contingency shuffling method. We construct pseudo contingencies of equal size to our actual contingencies but with random conditions. **(b,c)** Reshuffling test for proportion of contingency explained compared to shuffled pseudo-contingencies from **(a)**. The red line represents the mean variance explained by contingency for unit states in the RNN(RCN) model, while the grey bars represent the same measure for pseudo contingency replications. The RNN had significantly more variance explained by contingency than would be expected by chance while the RCN model did not. **(d)** Ratio of variance explained by contingency compared to mean pseudo-contingency shuffling, over 20 replicate RNNs. Dashed line represents chance level.

4.3 Discussion

In this study we defined a representational schema for delay tasks in which network states encode the mapping between expected stimuli and actions termed the contingency representation. We developed a novel task, the conditional delayed logic (CDL) task, capable of

dissociating contingency representations from stimulus or response representations. Human participants performing the CDL task demonstrated inter-conditional RT variance consistent with a contingency based strategy, and largely inconsistent with a stimulus memory strategy. In a trained recurrent neural network (RNN) we identified contingency representations and validated the functional role of contingency in the task. The structure of contingency representations in the model naturally captures experimentally observed phenomena including context-dependent tuning, mixed selectivity and dynamic coding for working memory. Lastly we generated falsifiable predictions for neural recordings of animals or humans performing the CDL task and compared results to alternative models.

Contingency representations can potentially unify memory, planning and cognitive control in a manner that does not require an external system to control the interactions between these subprocesses. In delay tasks where cues have a one-to-one correspondence with contingencies, the contingency representation is essentially indistinguishable from a sensory representation. In tasks where responses can be directly inferred from pre-delay cues, contingency states divide along future motor responses. Interestingly, tasks such as the CDL can exhibit both of the above cases. This directly links working memory to cognitive control through representational structure rather than positing an exogenous controller of inputs as in prior modeling[11]. Our behavioral data shows that humans can update their internal plans on the fly, between conditions of open contingency and closed contingency.

One intriguing property of contingency representations is that they can help to unify differing interpretations of neural activity in working memory delay tasks. Prior studies using neuronal recordings from primate association cortex have sought to dissociate retrospective sensory and more prospective signals for upcoming responses or expected stimuli[17, 37, 38], and found tuning for task rules[37]. Using our RNN model, and top-down theory generated from the contingency subspace of the CDL task, we offer a putative

explanation for some of the diversity of signals observed. By analyzing the network as it performed pairs of tasks with different geometric relationships in the contingency subspace, we found that units can appear tuned to rule, stimulus or action. This is due to the fact that the nonlinear transformation from rules and Cue A leads to different forms of collinearity in the contingency subspace.

The hypothesis that contingency could explain these results is bolstered by our behavioral results indicating that response times in our delay task are well explained by contingency. Response times have been shown to reflect cognitive processes[39], including difference in preparation and intention[43]. RT measurements have been related closely to neural activity across cortex[10, 46]. The observation that RT was strongly modulated by condition (closed vs. open) and that idiosyncratic RT variance could be explained by contingency in the CDL task provides evidence that participants utilized contingency during performance of the task.

Empirical analysis of unbiased samples of neurons have found that substantial populations are tuned to multiple features in a task. This phenomena has been termed mixed selectivity, with one hypothesis being that mixed selectivity generates a high dimensional and mostly unstructured basis to facilitate flexible computation [41, 4]. Here we find that since many nonlinear combinations of rule and stimuli can lead to the same contingency state, units tuned to contingency can appear mixed. While we do find some units in our RNN are tuned to the Cue A and rule, units are consistently better tuned to contingency and therefore would appear as mixed to traditional linear analysis methods. This contingency tuning, however, is highly structured. Our finding is therefore better matched to recent research exploring the trade-off between generalization and flexibility as a function of between-condition structure [6].

Recent studies have indicated there is substantial temporal dynamics in the pattern of activity seen during cue onset and early delay compared to late delay [50, 33, 49, 34]. One

hypothesis is that the mechanisms of neural persistence are intrinsically dynamic [5]. In our RNN, representations demonstrate what appears to be dynamics as the network moves from a stimulus dominant representation in the cue epoch to a contingency representation by the late delay epoch. In many experimental contexts contingencies are correlated with stimuli. In such cases the stimuli to contingency dynamics may appear as a change or rotation in stimulus coding (**Fig. 4.3**).

One important advantage to utilizing task-trainedRNN modeling in this study is that we do not impose any specific solution, thus allowing for emergent phenomena constrained by the task itself [3, 56]. This strongly contrasts with most of prior working memory modeling in which the structure of delay states is pre-specified by the researchers [8, 27]. Further observability and controllability over the RNN during information processing enabled us to implement precise perturbations to measure the causal role of contingency representations in behavior. These features enabled us to expand on recent work identifying shared modules in a trained multi-task network performing many tasks previously identified with prefrontal activity [56]. In complement to the heuristic modularity they observed, the precision of the CDL task in sharing common computational intermediate stages allowed us to produce a top down theory of modularity that we could then use to guide investigation and measure exact modular overlap in our RNN.

Our study opens substantial theoretical and experimental questions for future research. One important question is whether differences in task demands would lead to alternative strategies to contingency being preferred. For tasks like CDL in which stimulus dimensionality is higher than response dimensionality, the overall complexity of the problem can be reduced by computing through contingencies rather than stimuli. For tasks with higher response dimensionality than stimulus dimensionality, however, it is unclear whether contingency representations would be advantageous. Furthermore, contingency is not always invertible, and therefore may be unsuitable in contexts in which it is often necessary to

return to the previous cue. Of note, while contingency does reduce the CDL problem complexity, it does not reduce the amount of information stored mnemonically. To study these issues, the CDL task could be generalized to arbitrary dimensionalities of cue and action spaces, to systematically examine how capacity in both memory and problem complexity impacts strategies in human participants and in task-trained artificial neural network models.

Future experiments can record neural activity patterns from human participants or animals performing CDL tasks, to test whether internal representations are governed by contingency[21, 25, 48, 32]. Such an experiment could also help localize representational heterogeneity among cortical regions during the CDL task. Lastly, experiments could explore how changes in information flow between sensory, frontal and motor areas accompany the differential information routing for which we found behavioral evidence[45]. In turn, future modeling can incorporate aspects of known neurobiology which are potentially important for neural circuit computation, to extend beyond the relatively simple architecture used here. Neurobiologically motivated properties such as Dale's principle[47, 14], short-term synaptic plasticity[29], multi-regional connectivity constraints[36], or attentional mechanisms could be explored for their effects on the emergence or structure of contingency representations.

In conclusion, we introduced a representational schema, contingency representations, capable of unifying working memory and planning without use of an external controller. We found evidence of these representations in human behavior. In a task-trained neural network model we identified ways in which this representation can explain results on tuning from neurophysiological experiments on working memory, as well as provide new testable hypotheses for future studies of neural activity during cognitive tasks.

4.4 Methods

Behavioral task description

On each trial of the behavioral CDL task, the participant is shown two transient binary stimuli separated by a delay in addition to a rule cue on throughout the entirety of the trial. The participant is tasked with using the rule and stimuli to determine and report the correct response (**Figs. 4.2a**).

To increase statistical power and reduce training difficulty, five rules were selected from the ten original CDL rules (**Fig. 4.2a**): OR, AND, XOR, Memory (MEM) and Anti-Report (AREP). The OR, AND and XOR subtasks follow the stated Boolean logical operations. For the MEM subtask, the participant reports the identity of the first stimulus, Cue A (i.e., the stimulus that appears before the delay). For the AREP subtask, participants report the opposite of the second stimuli, Cue B (i.e., the stimulus that appears after the delay). For ease of understanding, and to reduce effects of differences in familiarity with logical operations, we displayed the five rules to participants as ”Either” (OR), ”Both” (AND), ”Only” (XOR), ”Memory” (MEM), and ”Reverse” (AREP) rather than their logical labels (**Fig. 4.2a**).

The five rules selected for the behavioral task were chosen such that there were 2 conditions each of [0-0] and [1-1] contingency, and 3 each of [0-1] and [1-0] contingency. This helped ensure no imbalance in the proportions of trials between the two closed contingencies nor between the two open contingencies. Further, since there was one more condition for each of the open contingencies the shorter response time for open trials could not be explained by frequency.

Preceding Cue A was a foreperiod of 800 ms in which the participant was shown only the rule cue in the upper center of the screen. The rule cue would then remain on the

screen throughout the entire trial until feedback. Following the foreperiod, the participant was shown Cue A, as either a ”0” or ”1”, in the center of the screen below the rule cue, for 500 ms. After Cue A was removed, only the rule was visible during a delay of 2000 ms. Cue B was then shown, as either a ”0” or ”1”, in the same location as used to display Cue A. The participant was able to respond by key stroke, with either a ”left arrow” for ”0” or a ”right arrow” for ”1”, at any point following Cue B onset. If the participant failed to make a choice within 2000 ms after Cue B onset, the trial would end and the participant would see a message telling them they timed-out, for a duration of 3000 ms. If the participant responded they would see either a ”Error” or ”Correct” feedback on the screen for 500 ms before the next trial begins. For each trial, rule, Cue A, and Cue B were selected randomly and independently.

The response time (RT) was calculated as the time between Cue B onset and key stroke response. Prior to the task, participants were instructed to respond as ”quickly and accurately” as possible, and that they could respond as soon as the second stimulus (Cue B) appeared on screen. In order to control for any bias either in response speed for ”left arrow” vs. ”right arrow” or bias in response to ”0” and ”1” stimuli, participants were randomly assigned a binary ”key mapping variable” which would either run the task as described above, or swap response mappings (i.e., left vs. right arrows to report responses) as well as ”0” and ”1” stimuli mappings.

Each participant completed 6 blocks of 64 trials each. The task was implemented through the PsychoPy package[35] in Python, and run on a Macbook Pro laptop.

Behavioral collection

18 participants were recruited from the general population between the ages of 18 and 35 in the New Haven, CT area. All participants completed and signed an informed consent approved by the Yale Institutional Review Board (IRB) and were paid for their participa-

tion. We used flyers and online advertising as our main recruitment methods. Of our 18 participants, 11 were female. The mean age was 24.3 years (standard deviation 4.2 years). Participants on average had 17.4 years of education (standard deviation 2.6 years).

Participants who contacted the lab were given a brief phone screening for eligibility. Participants with psychiatric diagnosis or history were excluded from the sample, as well as participants who had substantially impaired and uncorrected eyesight and participants with less than a 5th-grade reading level.

Eligible participants were invited into laboratory testing room, asked to read and sign an experimental consent form and then a brief demographic survey prior to task training. Following this they were shown a brief task description that explained the task, including the order of events, the goal, and the task. Participants were informed that their reward would be a function of accuracy and response time but were not told an exact calculation for reward.

Participants were then given a training block of 40 randomly selected trials of the CDL task. Participants who failed to get greater than 90% accuracy were required to repeat the training block. If the training block was failed three times the participant was paid the base compensation and excluded from the study. Only one participant was excluded this way.

Participants then performed 6 blocks of 64 trials, after each block the participant was able to take a self timed break. At the end of the task, their bonus reward was calculated based on their RTs and accuracy according to:

$$\text{Reward} = \min\left(\text{round}\left(\frac{15}{1+e^{0.25*(n_{err}-5)}}\right)\right) + \min\left(\text{round}\left(\frac{15}{1+e^{10*(\mu_{rt}-0.5)}}\right)\right)$$

where n_{err} represents the total number of errors and μ_{rt} represents the average RT (in seconds) for correct trials. Reward was paid in US dollars at the end of the experiment, following a brief demographic survey.

Behavioral analysis

To test the differences between closed and open trials, we averaged each participant's RTs by closure, averaging all open trials and all closed trials, and then conducted a paired two-sided t-test across participants. Similarly, to test the significance of closure on conditionally closed trials, we averaged the RTs of each condition (Cue A $\times$ rule combination) separately. Then we ran a two-sided paired t-test between mean Cue A=0 RTs and Cue A=1 RTs across participants within each task. We tested the individual-participant significance by using a two-sample t-test to directly test the significance of the difference between Cue A=0 and Cue A=1 trials, for each rule independently, for each participant (**Fig. 4.2c**).

To examine learning effects, we averaged a given participant's RT by condition (Cue A $\times$ rule) for the first two blocks which we termed "early" trials, and the last two blocks which we termed "late" trials. We determined the closure effect for OR and AND trials as the difference, for that participant, in mean RT in the open condition and the closed condition of those tasks. We termed this difference the closure effect. Then we calculated a two-sided paired t-test to measure whether this closure effect had increased from early to late trials, across the group (**Fig. 4.2d**).

To examine the effect of contingency on behavior, we measured the variance in trial-wise RT that is explainable by contingency. For each participant, we fit a linear model on correct trials, with RT as the predicted variable and the predictors being contingency, closure, rule, target, and congruency. Contingency was the contingency that would be applicable during that trial, based on the rule and Cue A. Rule is either XOR, OR, AND, Memory or Reverse (Anti-Report) depending on trial. Target was the correct response for that trial. Lastly, congruency is a binary variable representing whether Cue B was the same as the target. Congruency was used to control for possible effects of pro-/anti-match bias

in RTs. Then we fit a second linear model with all the predictors above except contingency. Each predictor was encoded as a one-hot and models were fit with an L2 regularization penalty of 0.01.

Using these models we then calculated the percent of RT variance explained by each model. The difference between the full model and the model lacking a contingency predictor we termed the "ΔEV", representing the amount of additional variance explained by accounting for contingency.

We tested for statistical significance using a permutation analysis, in which we shuffled trial contingency labels 1000 times and recalculated ΔEV, forming a null distribution against which we conducted hypothesis testing. 53% of participants (9 of 17) had a ΔEV greater than 95% of permuted samples. We then used a binomial test to identify the significance of the proportion of participants for whom contingency was found to be a significant predictor (**Fig. 4.2e**).

We calculated the variance attributable to contingency using the models below. Model 1 includes contingency as a predictor, and is specified by:

$$RT_{1,i} = w_0 + w_{\text{Clos}}\ \delta_i^{\text{Clos}} + \sum_{c \in \{\text{contingencies}\}} w_c\ \delta_{i,c}^{\text{Cont}} + \sum_{r \in \{\text{rules}\}} w_r\ \delta_{i,r}^{\text{Rule}} + w_{\text{Targ}}\ \delta_i^{\text{Targ}} + w_{\text{Cong}}\ \delta_i^{\text{Cong}} \tag{4.1}$$

where RT_i is the response time on trial i, and regression dummy variables are defined as Kronecker delta functions for the match of conditions in trial i. $\delta_{i,c}^{\text{cont}}$ is 1 when the contingency of trial i matches contingency c and 0 otherwise. δ_i^{clos} is 1 or 0 when the trial's closure condition is closed or open, respectively. $\delta_{i,r}^{\text{rule}}$ is 1 when the rule of trial i matches rule r. δ_i^{targ} is 1 or 0 when the trial's target condition is 1 or 0, respectively. δ_i^{cong} is 1 or 0 when the trial's congruency condition is congruent or incongruent, respectively.

Model 2 includes all predictors from Model 1 except for contingency:

$$RT_{2,i} = w_0 + w_{\text{Clos}}\ \delta_i^{\text{Clos}} + \sum_{r\in\{\text{rules}\}} w_r\ \delta_{i,r}^{\text{Rule}} + w_{\text{Targ}}\ \delta_i^{\text{Targ}} + w_{\text{Cong}}\ \delta_i^{\text{Cong}} \tag{4.2}$$

Here Model 1 has 12 parameters (1 constant, 4 for contingency, 1 for closure, 5 for rule, 1 for target, and 1 for congruency), and Model 2 has 8 parameters.

We quantified the sum of squared errors (SSE) for each model:

$$\text{SSE}_1 = \sum_i \left(RT_i - \hat{RT}_{1,i}\right)^2 \tag{4.3}$$

$$\text{SSE}_2 = \sum_i \left(RT_i - \hat{RT}_{2,i}\right)^2 \tag{4.4}$$

where the sum is over trials, and $\hat{RT}_{m,i}$ is the predicted RT for trial i from model m. The proportion of explained variance (EV) is given by:

$$\text{EV}_1 = 1 - \frac{\text{SSE}_1}{\text{SST}} \tag{4.5}$$

$$\text{EV}_2 = 1 - \frac{\text{SSE}_2}{\text{SST}} \tag{4.6}$$

where SST is the total sum of squares (variance) of RT across trials. Then the amount of extra explained variance from including contingency regressors is $\Delta\text{EV} = \text{EV}_1 - \text{EV}_2$.

RNN model architecture

All RNN experiments were performed on continuous-time recurrent neural networks discretized to a 10-ms time step. RNNs contained 200 recurrently connected rectified-linear units (ReLU) in the hidden layer. Recurrent units received input as a linear combination of network inputs. Outputs were linear readouts of rectified recurrent unit activity. The

network was governed by the following equations:

$$\tau \dot{x} = -x + W^{\text{rec}} r + W^{\text{in}} u + b^{\text{rec}} + \eta \tag{4.7}$$

$$r = f(x) = \max(x, 0) \tag{4.8}$$

$$z = W^{\text{out}} r + b^{\text{out}} \tag{4.9}$$

where W_{rec} is the recurrent weight matrix, W_{in} is the input weight matrix, and W_{out} is the output weight matrix. x is the unit state vector, r is the unit activity vector, u is the vector of inputs at a given time, and z is the output vector. b_{rec} and b_{out} are constant biases into the recurrent and output nodes respectively. τ = 100 ms is the internal time constant of units and η is gaussian noise added to the recurrent layer. Lastly, $f(x)$ is the ReLU positive linear rectification function.

All inputs and outputs were represented using one-hot encodings. The network received cue input from 4 input channels, with the first two being assigned to represent a binary input during the first stimulus epoch, Cue A, and the latter two representing binary input in the second stimulus epoch, Cue B. The network received rule input through 10 channels with each channel associated with the instructed subtask being on and all others off[28, 56]. The network response was recorded from two output nodes, each representing one of the two responses. Final response was determined by the output node with the greatest activity at the final time-point.

RNN task description

On each trial the network received as inputs two binary cues and was tasked with choosing the correct response, as designated by a context cue presented tonically through the task. The X(N)OR, (N)OR, and (N)AND tasks followed the logic of their rules as applied to binary stimuli. For the Memory (MEM) and Anti-Memory (AMEM) tasks the network had

to respond with the first cue or the opposite of the first cue, respectively. Similarly, for the Report (REP) and Anti-Report (AREP) tasks the network had to respond with the second cue or the opposite of the second cue, respectively. All context-cue pair contingencies are detailed in **Fig. 4.1b**.

Preceding Cue A was a foreperiod of 200 ms in which only the rule input was on. The rule input is presented throughout the trial. Following the foreperiod, one of two Cue A channels was set to 1 for 100 ms. Next both Cue A channels were returned to 0 for a delay period of 2000 ms. After the delay one of the two Cue B channels was set to 1 for 100 ms. Following Cue B onset the network was instructed to produce a steady output of 1 in the output channel corresponding to the correct response for that trial. Output was masked until 100 ms following Cue B, such that output magnitudes prior to that time were not considered in the loss function (and therefore unconstrained).

RNN training

The RNN's hidden (recurrent) weights were initialized as a random gaussian matrix with spectral radius equal to 1.1, and the initial input and output weights were initialized with the Xavier method[19]. RNNs were trained using an Adam optimizer[23] as implemented in TensorFlow 1.11.0, and with regularization on weights and unit activity[51, 56]. Input, recurrent, and output weight matrices were regularized with L1 penalties of 0.01, and unit activity was regularized with an L2 penalty of 1. The overall loss function is thereby given by:

$$\text{Loss} = |z_{\text{targ}} - z|^2 + |f(x)|^2 + 0.01\left|W^{\text{in}}\right| + 0.01\left|W^{\text{rec}}\right| + 0.01\left|W^{out}\right| \tag{4.10}$$

where z_{targ} is the target output.

The task was trained and tested with noise injected into input channels and recurrent

units. Gaussian noise ($\mu = 0$, $\sigma = 0.01$) inserted into each stimulus channel, and into each recurrent unit ($\mu = 0$, $\sigma = 0.1$), at each time point.

Networks were trained with a curriculum learning regime in which the memory delay duration was progressively extended over training[14]. Initially the network was trained for 2,000,000 trials with a delay duration of 400 ms. This was followed by training the network for 200,000 trials on increasing delays of 200 ms greater than the previous iteration. When the delay length reached 2000 ms the network was trained again for a final 2,000,000 trials. Training proceeded with batches of 128 trials.

To test for robustness of results in RNNs across initial conditions and training histories, 20 replicates of the network were trained, each starting from a different random initialization.

RNN Analysis

Explained variance analysis. We utilized a linear model with binary coded contingency as predictors to identify the proportion of across-trial state variance explained by each contingency for each unit j in our network:

$$r_j = w_j^0 + \sum_{c \in \{\text{contingencies}\}} w_j^c \; \delta_{i,c}^{\text{Cont}} \tag{4.11}$$

where, as in the empirical analysis, $\delta_{i,c}^{\text{cont}}$ is 1 when the contingency of trial i matches contingency c and 0 otherwise. This regression model thereby uses 5 parameters per unit (1 baseline and 4 contingency). The proportion of explained variance can be calculated per unit j: $\text{EV}_j = 1 - \text{SSE}_j / \text{SST}_j$.

We defined the late delay epoch state as the dynamic state variable of each unit at the time step prior to onset of Cue B. For demonstration of an example unit, **Fig. 4.4a** plots the mean rectified state variable, averaged across trials for each rule and Cue A condition,

for the recurrent unit that had the most [0-1] contingency variance.

In order to measure differences in sample and delay tuning we utilized linear models to calculate, separately for each unit, the amount of state variance across trials explained by Cue A identity and contingency. For **Fig. 4.4c**, variance explained was calculated at two time points: the final time point before Cue A offset (Sample) and the time point prior to Cue B onset (Delay). Explained variance was calculated as below, and plotted is the average variance explained across units for each feature and time point.

Model 1 used contingencies (Eq. 4.11). Model 2 used Cue A identities:

$$r_j = w_j^0 + \sum_{a \in \{\text{Cue A stimuli}\}} w_j^a \, \delta_{i,a}^{\text{Cue-A}} \tag{4.12}$$

The explained variance for given feature f (contingency or Cue A) is given by:

$$\text{EV}^f = 1 - \frac{\text{SSE}^f}{SST^f} \tag{4.13}$$

These regression models and calculation of explained variance can be performed in a time-resolved manner to characterize the time course of coding in the network (**Fig. 4.4b,c**).

For comparison to the experimental results from Rainer et al. (1999)[38], we utilized the WebPlotDigitizer[42] to replicate plotted data values (**Fig. 4.4d**).

Dynamic coding axis. For **Fig. 4.4e**, at each time point, and separately for each rule, we calculated the first principal component (PC) of network states across trials, following prior experimental studies[33, 44, 34]. We then measured the angle between first PCs from each pair of time points identifying the degree to which the dominant coding axis had shifted between time points[44]. We then plotted the angular similarity between time points averaged across rules.

Dynamic decoding. For **Fig. 4.4f**, we first separated the data into two equal halves of 500

trials as train and test sets. We trained a linear kernel support vector machine (SVM), for each rule to classify Cue A on the stimulus epoch activity of the train set (sample decoder in **Fig. 4.4f**), and then tested it on each time point throughout the task on the test set. Next we trained a separate linear SVM to classify Cue A at each time point on the train set, testing it on the test set trials from the corresponding time point (dynamic decoder in **Fig. 4.4f**). This analysis is analogous to dynamic decoding analyses in prior experimental studies examining cross-temporal generalization of working memory representations in prefrontal cortex[50, 49, 9]. Plotted is mean SVM accuracy averaged across rules.

Contingency subspace. We determined the first axis of the contingency subspace via a linear regression to identify an axis which could decode the expected response to an upcoming Cue B=0, based on the Cue A and rule on that trial, using the late delay epoch unit activation. We similarly identified the second axis through a regression for the expected response to an upcoming Cue B=1. While axes were determined independently, we found they were always nearly orthogonal over 20 replicate RNNs with a mean between axis angle of 92.6 degrees (sd 4.7 degrees). Further the regression method caused a modest rescaling of the original space with a mean axis norm of 1.4 (sd 0.17).

To get the contingency representation we projected the state vector from individual trials into the plane identified. To avoid any issues with overfitting, we fit the regression on half our data which we used as a training set, and utilized the held-out half of our data for figures and analysis.

Variance captured by subspaces. In order to test the hypothesis that more condition-wise variance is explained by the contingency subspace than would be expected by chance, we examined whether random two dimensional subspaces would contain equal variance (**Fig.= 4.3c**). First we generated random orthogonal planes using Gram-Schmidt orthogonalization and rescaling, normalized such that the axes were equal in norm to the contingency subspace axes. We then orthogonalized the two contingency subspace axes and

measured the sum of the state variance in the contingency plane (the red vertical line). We compared the variance in the contingency subspace to the state variance across the 20,000 randomly sampled orthogonal planes (**Fig. 4.3c**). We then ran this analysis over our 20 replicate networks and calculated a normalized variance measure, defined as the variance in the contingency subspace divided by the mean of the variance across random orthogonal planes (**Fig. 4.3d**).

Perturbation analysis. To functionally identify the relevance of the contingency subspace for behavior we applied a state-space perturbation analysis. A random vector was added to the state of the network at the time point immediately preceding Cue B onset. This perturbation was either linearly dependent on the axes of the contingency subspace and therefore lie within the subspace, or orthogonal to the subspace. We tested random perturbations of increasing vector norm (magnitude) and measured the resulting decrease in accuracy. We repeated this analysis for our 20 replicate RNNs. Plotted are lines for the mean accuracy across replicates for a given perturbation either in the subspace or out of the subspace with the band representing the 90% confidence interval (**Fig. 4.3e**, **Supplementary Fig. 4.S4**).

Singular value decomposition (SVD) analysis of CDL task

First we divide the CDL task into (i) pre-delay features, i.e., features known to the agent prior to the delay, and (ii) post-delay features. Pre-delay features include the rule and Cue A, while post-delay features include the Cue B and response. We represent each feature as a one-hot encoding. Then we can build an interaction matrix for Cue A $\times$ Rule and a separate interaction matrix for Cue B $\times$ response. Finally we construct the correlation matrix between our two interaction matrices, and factorize it with a singular value decomposition (SVD). This results in a U matrix which represents loadings for our task conditions, a V matrix that represents loadings onto our contingencies, and a Σ matrix

that represents the weighting of those loadings.

$$F_{\text{pre}} = \text{Cue A} \times \text{Rule}. \tag{4.14}$$

$$F_{\text{post}} = \text{Cue B} \times \text{Response} \tag{4.15}$$

$$M = \rho(F_{\text{pre}}, F_{\text{post}}) \tag{4.16}$$

$$M = U \cdot S \cdot V \tag{4.17}$$

Results of this SVD analysis are shown in **Supplementary Fig. 4.S1**.

Task pair analysis

To identify Cue A tuning in the contingency subspace we first mapped each Cue A $\times$ Rule condition of a given pair of tasks to its associated contingency state. Here we use an idealized set of orthonormal contingency coding axes which provide the only tuning in the system, with the first axis encoding the response if Cue B=0, and a second axis encoding the response if Cue B=1 (**Fig. 4.5a**).

Each task pair is represented four possible states, 2 Cue A states $\times$ 2 rule states. For each pair we first averaged states across rules . We took the measured Euclidean distance between Cue A=0 mean and the Cue A=1 mean as our theoretical prediction of cue tuning. Similarly, to identify rule tuning we averaged across Cue A states. The Euclidean distance between the means for the two rules in the contingency subspace was taken as our theoretical prediction of rule tuning. We repeated this procedure through all 100 possible pairs of rules (including each rule paired with itself) to generate the full matrix of predictions. We then defined a theoretical cue:rule tuning ratio as the ratio of our predicted cue tuning over predicted rule tuning (**Fig. 4.5b**).

In each of 20 replicate RNNs trained on the CDL task, we isolated trials belonging to a pair of tasks. We then averaged the state vectors by condition (Cue A $\times$ Rule pairs). As

above we then averaged the state of the network and measured the Euclidean distance in state space across rule to get cue tuning and across cue to get rule tuning. We repeated this procedure for all task pairs as above. We then divided our measured cue tuning by rule tuning to get a RNN rule to cue tuning ratio. Finally we averaged across our 20 replicates. To compare the RNNs to theoretical predictions, we used a Spearman rank correlation test to identify similarities in the pattern of the lower triangular matrices of both the predicted and measured cue to rule tuning ratios (**Fig. 4.5c**).

Comparison models

RCN model. We implemented a randomly connected network (RCN) model[4, 26] to investigate differential predictions between unstructured-high dimensional nonlinear representations and our task-trained RNN models. To generate internal representations for RCN models, we generated one-hot representations of rule, Cue A, and Cue B for each trial, which we term u_{inst} of dimensionality 1000×14. Then we multiply these representations by a matrix, W_{rand}, of dimension 14×200, the number of input variables by the maximal dimensionality of the 200 recurrent units of the RNN. Elements of W_{rand} were drawn from a random normal distribution. A threshold variable θ for each element drawn uniformly from the interval $[-1, 1]$ were then subtracted. Finally we transform the representation with a sign operation such that values are represented by ± 1.

$$\text{RCN} = \text{sign}(u_{\text{inst}} \cdot W_{\text{rand}} - \theta) \tag{4.18}$$

For the inputs model we took the one-hot representations of rule, Cue A and Cue B, u_{inst}, and simply ran analysis directly on these representations.

Representational similarity analysis (RSA). To compare our trained RNN to the RCN model in representational structure, we applied a representational similarity analysis to

both networks[25]. For the RNN we took the activity vector at the time point prior to Cue B onset. We generated representational similarity matrices by averaging activity across trials for each Cue A $\times$ rule condition, and then taking the pairwise Pearson correlation between the averaged activity vectors (**Fig. 4.6a,b**).

We then compared these matrices to three binarized candidate representational matrices: (i) Cue, with two conditions being similar if they had the same Cue A; (ii) rule, with two conditions being similar if they shared a rule; and (iii) contingency, with two conditions being similar if they share a contingency (**Fig. 4.6c**). We repeated this analysis for 20 randomly constructed RCNs and our 20 replicate RNNs. Plotted is the distributions over these replicates separated for cue, rule and contingency, respectively (**Fig. 4.6d**).

Variance explained. In order to identify the proportion of variance explained by Cue A, rule, and contingency in each model, we fit a linear model for each feature to each unit states across trials using a regularized least squares ($\lambda = 0.1$). For the RNN, we used the state at the time step prior to Cue B onset. We then calculated the proportion of variance explained as $1 - \mathrm{SSE}/\mathrm{SST}$. We then conducted this analysis for each unit, taking the average over a given network. Shown is the distribution of mean unit variance explained by that feature over 20 RCN replicates and 20 RNN replicates (**Fig. 4.6e**).

Dimensionality. To measure effective dimensionality of states in our models, we used the participation ratio[18, 15] calculated as $\left(\Sigma_i \bar{\lambda_i}^2\right)^{-1}$, where $\bar{\lambda_i} = \lambda / \Sigma_i \lambda_i$ and $\{\lambda_i\}$ are the eigenvalues of the covariance matrix. We applied this measure to the RNN states at time step prior to Cue B onset. We also used this analysis to measure the effective dimensionality of the RCN activity and of the input data itself as comparisons (**Fig. 4.6f**).

Cross-context generalization. We utilized a cross-condition generalization (CCG) analysis[6] to test the extent to which the geometry of the contingency subspace represented common information between different trial conditions with the same contingency state. We fit the contingency subspace, as described above, except we held out all trials with a given rule–

Cue A condition. We then projected the held-out trials into the subspace and used the four quadrants of the contingency subspace as a classifier, with each trial being classified as the contingency state of the appropriate quadrant. We repeated this analysis for our 20 RNN replicates, as well as for our 20 RCN replicates. Plotted is the distribution of accuracies for the CCG classifier across networks (**Fig. 4.6g**).

Partitioned variance analysis

To measure the extent to which contingency was over-represented in our RNN model compared to statistically matched nonlinear structure, we implemented a permutation test and identified mean unit state variance explained. Then we permuted contingencies between Cue A $\times$ rule pairs to generate pseudo-contingencies (e.g. all Cue A=0, Rule=AND trials were assigned the same new random contingency) (**Fig. 4.7a**). We regressed out linear effects of Cue A and rule, fitting a linear model and taking the residual for further analysis. Then we used our pseudo-contingency labels to fit a linear model using regularized least squares ($\lambda = 0.1$). Finally we calculated variance explained as $\frac{\mathrm{SSE}_{\mathrm{cont}}}{SST}$. We compared the actual contingency variance explained to 1000 randomly sampled pseudo-contingency sets (**Fig. 4.7b**).

Specifically, we perform the following regression model:

$$r = w^0 + \sum_{a \in \{\text{Cue A stimuli}\}} w^a \; \delta_{i,a}^{\text{Cue-A}} + \sum_{r \in \{\text{rules}\}} w_r \; \delta_{i,r}^{\text{Rule}} \tag{4.19}$$

We define the residuals of this regression model:

$$\text{res} = r - \hat{r} \tag{4.20}$$

where $\hat{r}$ is the predicted activity for each trial. We next perform a second regression on the

residual:

$$\text{res} = w^0 + \sum_{c \in \{\text{(pseudo-)contingencies}\}} w_c \, \delta_{i,c}^{\text{Cont}} \tag{4.21}$$

using either the true contingencies, or the shuffled pseudo-contingencies to generate the null distribution.

We repeated this analysis for the RCN data, as well as all replicates of the RNN (**Fig. 4.7b,c**). We calculated normalized variance as the mean variance explained by contingency labels over the mean variance explained by pseudo-contingency permutations. Shown is the distribution across RNN replicates (**Fig. 4.7d**).

Bibliography

[1] Christiane Ahlheim and Bradley C Love. Estimating the functional dimensionality of neural representations. *Neuroimage*, 179:51–62, 10 2018. 52

[2] David Badre, Apoorva Bhandari, Haley Keglovits, and Atsushi Kikumoto. The dimensionality of neural representations for control. *Curr Opin Behav Sci*, 38:20–28, Apr 2021. 52

[3] Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Curr Opin Neurobiol*, 46:1–6, 10 2017. 37, 59

[4] Omri Barak, Mattia Rigotti, and Stefano Fusi. The sparseness of mixed selectivity neurons controls the generalization-discrimination trade-off. *J Neurosci*, 33(9):3844–56, Feb 2013. 51, 58, 74

[5] Omri Barak, David Sussillo, Ranulfo Romo, Misha Tsodyks, and L F Abbott. From fixed points to chaos: three models of delayed discrimination. *Prog Neurobiol*, 103:214–22, Apr 2013. 59

[6] Silvia Bernardi, Marcus K Benna, Mattia Rigotti, Jérôme Munuera, Stefano Fusi, and C Daniel Salzman. The geometry of abstraction in the hippocampus and prefrontal cortex. *Cell*, 183(4):954–967.e21, Nov 2020. 54, 58, 75

[7] Flora Bouchacourt and Timothy J Buschman. A flexible model of working memory. *Neuron*, 103(1):147–160.e8, 07 2019. 37

[8] N. Brunel and X. J. Wang. Effects of neuromodulation in a cortical network model of object working memory dominated by recurrent inhibition. *J Comput Neurosci*, 11(1):63–85, 2001. 37, 59

[9] Sean E Cavanagh, John P Towers, Joni D Wallis, Laurence T Hunt, and Steven W Kennerley. Reconciling persistent and dynamic hypotheses of working memory coding in prefrontal cortex. *Nat Commun*, 9(1):3498, 08 2018. 49, 71

[10] Mark M Churchland, Byron M Yu, Stephen I Ryu, Gopal Santhanam, and Krishna V Shenoy. Neural variability in premotor cortex provides a signature of motor preparation. *J Neurosci*, 26(14):3697–712, Apr 2006. 58

[11] J D Cohen, D Servan-Schreiber, and J L McClelland. A parallel distributed processing approach to automaticity. *Am J Psychol*, 105(2):239–69, 1992. 57

[12] Christos Constantinidis and Emmanuel Procyk. The primate working memory networks. *Cogn Affect Behav Neurosci*, 4(4):444–465, Dec 2004. 36

[13] Christopher J Cueva, Alex Saez, Encarni Marcos, Aldo Genovesio, Mehrdad Jazayeri, Ranulfo Romo, C Daniel Salzman, Michael N Shadlen, and Stefano Fusi. Low-dimensional dynamics for working memory and time encoding. *Proc Natl Acad Sci U S A*, 117(37):23021–23032, 09 2020. 37

[14] Daniel B Ehrlich, Jasmine T Stone, David Brandfonbrener, Alexander Atanasov, and John D Murray. Psychrnn: An accessible and flexible python package for training recurrent neural network models on cognitive tasks. *eNeuro*, 8(1), 2021. 60, 69

[15] Matthew Farrell, Stefano Recanatesi, Timothy Moore, Guillaume Lajoie, and Eric Shea-Brown. Recurrent neural networks learn robust representations by dynamically balancing compression and expansion. *bioRxiv*, 2019. 52, 75

[16] S Funahashi, C J Bruce, and P S Goldman-Rakic. Mnemonic coding of visual space in the monkey's dorsolateral prefrontal cortex. *J Neurophysiol*, 61(2):331–49, Feb 1989. 36, 38

[17] S Funahashi, M V Chafee, and P S Goldman-Rakic. Prefrontal neuronal activity in rhesus monkeys performing a delayed anti-saccade task. *Nature*, 365(6448):753–6, Oct 1993. 40, 57

[18] Peiran Gao, Eric Trautmann, Byron Yu, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, and Surya Ganguli. A theory of multineuronal dimensionality, dynamics and measurement. *bioRxiv*, 2017. 75

[19] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. 68

[20] P S Goldman-Rakic. Cellular basis of working memory. *Neuron*, 14(3):477–85, Mar 1995. 36, 40

[21] John-Dylan Haynes and Geraint Rees. Decoding mental states from brain activity in humans. *Nat Rev Neurosci*, 7(7):523–34, Jul 2006. 60

[22] Laurence T Hunt, W M Nishantha Malalasekera, Archy O de Berker, Bruno Miranda, Simon F Farmer, Timothy E J Behrens, and Steven W Kennerley. Triple dissocia-

tion of attention and decision computations across prefrontal cortex. *Nat Neurosci*, 21(10):1471–1481, 10 2018. 52

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 68

[24] Nikolaus Kriegeskorte. Deep neural networks: A new framework for modeling biological vision and brain information processing. *Annu Rev Vis Sci*, 1:417–446, Nov 2015. 37

[25] Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. Representational similarity analysis - connecting the branches of systems neuroscience. *Front Syst Neurosci*, 2:4, 2008. 51, 52, 60, 75

[26] Grace W Lindsay, Mattia Rigotti, Melissa R Warden, Earl K Miller, and Stefano Fusi. Hebbian learning in a random network captures selectivity properties of the prefrontal cortex. *J Neurosci*, 37(45):11021–11036, 11 2017. 37, 51, 74

[27] Christian K Machens, Ranulfo Romo, and Carlos D Brody. Flexible control of mutual inhibition: a neural model of two-interval discrimination. *Science*, 307(5712):1121–4, Feb 2005. 59

[28] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, Nov 2013. 44, 67

[29] Nicolas Y Masse, Guangyu R Yang, H Francis Song, Xiao-Jing Wang, and David J Freedman. Circuit mechanisms for the maintenance and manipulation of information in working memory. *Nat Neurosci*, 22(7):1159–1167, 07 2019. 60

[30] Francesca Mastrogiuseppe and Srdjan Ostojic. Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron*, 99(3):609–623.e29, 08 2018. 37

[31] E K Miller, C A Erickson, and R Desimone. Neural mechanisms of visual working memory in prefrontal cortex of the macaque. *J Neurosci*, 16(16):5154–5167, Aug 1996. 38

[32] Robert M Mok and Bradley C Love. Abstract neural representations of category membership beyond information coding stimulus or response. *J Cogn Neurosci*, pages 1–17, Nov 2020. 60

[33] John D Murray, Alberto Bernacchia, Nicholas A Roy, Christos Constantinidis, Ranulfo Romo, and Xiao-Jing Wang. Stable population coding for working memory coexists with heterogeneous neural dynamics in prefrontal cortex. *Proc Natl Acad Sci U S A*, 114(2):394–399, Jan 2017. 58, 70

[34] Aishwarya Parthasarathy, Cheng Tang, Roger Herikstad, Loong Fah Cheong, Shih-Cheng Yen, and Camilo Libedinsky. Time-invariant working memory representations in the presence of code-morphing in the lateral prefrontal cortex. *Nat Commun*, 10(1):4995, 11 2019. 58, 70

[35] Jonathan Peirce, Jeremy R Gray, Sol Simpson, Michael MacAskill, Richard Höchenberger, Hiroyuki Sogo, Erik Kastman, and Jonas Kristoffer Lindeløv. Psychopy2: Experiments in behavior made easy. *Behav Res Methods*, 51(1):195–203, 02 2019. 62

[36] Lucas Pinto, Kanaka Rajan, Brian DePasquale, Stephan Y Thiberge, David W Tank, and Carlos D Brody. Task-dependent changes in the large-scale dynamics and necessity of cortical regions. *Neuron*, 104(4):810–824.e9, 11 2019. 60

[37] J Quintana and J M Fuster. From perception to action: temporal integrative functions of prefrontal and parietal neurons. *Cereb Cortex*, 9(3):213–21, 1999. 36, 37, 40, 57

[38] G Rainer, S C Rao, and E K Miller. Prospective coding for objects in primate prefrontal cortex. *J Neurosci*, 19(13):5493–505, Jul 1999. 37, 47, 48, 57, 70

[39] Roger Ratcliff and Gail McKoon. The diffusion decision model: theory and data for two-choice decision tasks. *Neural Comput*, 20(4):873–922, Apr 2008. 58

[40] Evan D Remington, Devika Narain, Eghbal A Hosseini, and Mehrdad Jazayeri. Flexible sensorimotor computations through rapid reconfiguration of cortical dynamics. *Neuron*, 98(5):1005–1019.e5, 06 2018. 37

[41] Mattia Rigotti, Omri Barak, Melissa R Warden, Xiao-Jing Wang, Nathaniel D Daw, Earl K Miller, and Stefano Fusi. The importance of mixed selectivity in complex cognitive tasks. *Nature*, 497(7451):585–90, May 2013. 37, 51, 58

[42] Ankit Rohatgi. Webplotdigitizer, July 2020. 70

[43] D A Rosenbaum. Human movement initiation: specification of arm, direction, and extent. *J Exp Psychol Gen*, 109(4):444–74, Dec 1980. 58

[44] Román Rossi-Pool, Antonio Zainos, Manuel Alvarez, Jerónimo Zizumbo, José Vergara, and Ranulfo Romo. Decoding a decision process in the neuronal population of dorsal premotor cortex. *Neuron*, 96(6):1432–1446.e7, 12 2017. 49, 70

[45] Markus Siegel, Timothy J Buschman, and Earl K Miller. Cortical information flow during flexible sensorimotor decisions. *Science*, 348(6241):1352–5, Jun 2015. 60

[46] Lawrence H Snyder, Anthony R Dickinson, and Jeffrey L Calton. Preparatory delay activity in the monkey parietal reach region predicts reach reaction times. *J Neurosci*, 26(40):10091–9, Oct 2006. 58

[47] H Francis Song, Guangyu R Yang, and Xiao-Jing Wang. Training excitatory-inhibitory recurrent neural networks for cognitive tasks: A simple and flexible framework. *PLoS Comput Biol*, 12(2):e1004792, Feb 2016. 60

[48] Chun Siong Soon, Anna Hanxi He, Stefan Bode, and John-Dylan Haynes. Predicting free choices for abstract intentions. *Proc Natl Acad Sci U S A*, 110(15):6217–22, Apr 2013. 60

[49] Eelke Spaak, Kei Watanabe, Shintaro Funahashi, and Mark G Stokes. Stable and dynamic coding for working memory in primate prefrontal cortex. *J Neurosci*, 37(27):6503–6516, 07 2017. 49, 58, 71

[50] Mark G Stokes, Makoto Kusunoki, Natasha Sigala, Hamed Nili, David Gaffan, and John Duncan. Dynamic coding for cognitive control in prefrontal cortex. *Neuron*, 78(2):364–75, Apr 2013. 58, 71

[51] David Sussillo, Mark M Churchland, Matthew T Kaufman, and Krishna V Shenoy. A neural network that finds a naturalistic solution for the production of muscle activity. *Nat Neurosci*, 18(7):1025–33, Jul 2015. 68

[52] J D Wallis, K C Anderson, and E K Miller. Single neurons in prefrontal cortex encode abstract rules. *Nature*, 411(6840):953–6, Jun 2001. 36, 37, 40

[53] Xiao-Jing Wang. Synaptic reverberation underlying mnemonic persistent activity. *Trends Neurosci*, 24(8):455–463, Aug 2001. 40

[54] Xiao-Jing Wang. Decision making in recurrent neuronal circuits. *Neuron*, 60(2):215–34, Oct 2008. 37

[55] Kong-Fatt Wong and Xiao-Jing Wang. A recurrent network mechanism of time integration in perceptual decisions. *J Neurosci*, 26(4):1314–28, Jan 2006. 51

[56] Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nat Neurosci*, 22(2):297–306, 02 2019. 37, 59, 67, 68

4.5 Appendix

Supplementary Figures

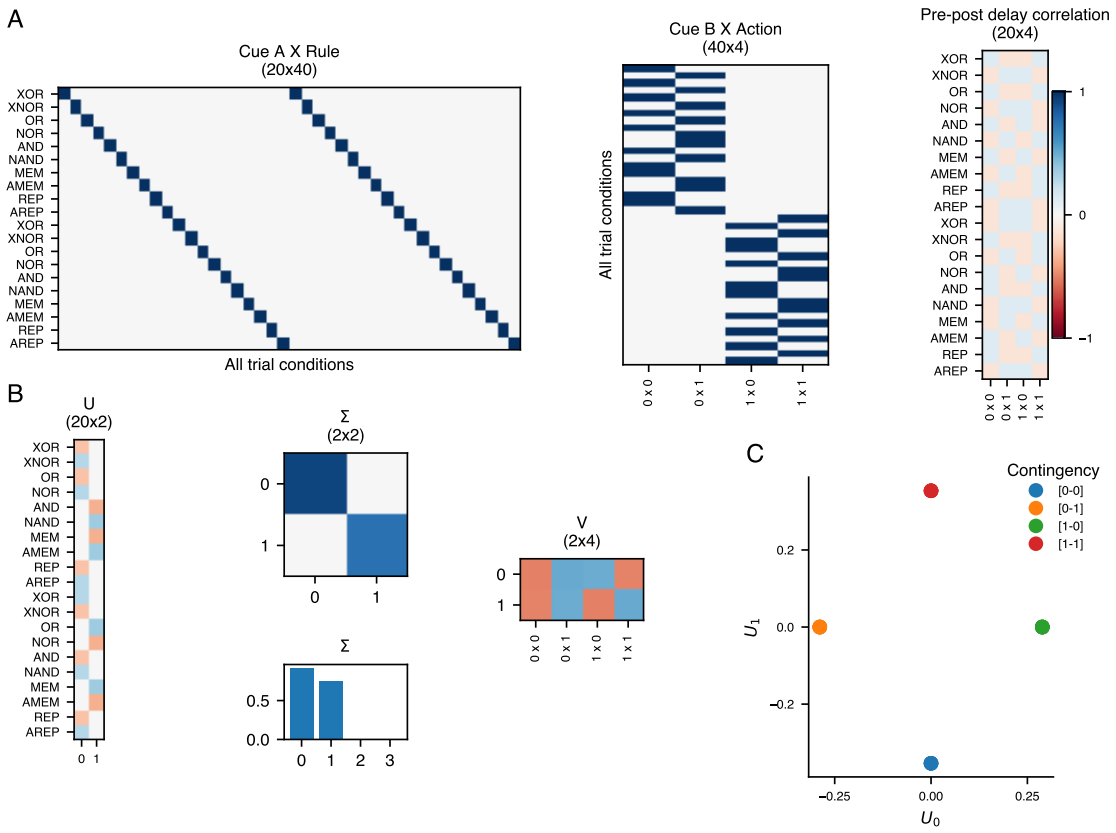


Figure 4.S1: **Singular value decomposition (SVD) analysis of CDL task. (a)** Left: Cue A by Rule interaction matrix (20 x 40), representing the interaction of all trial information provided prior to the delay, Middle: Cue B by Action interaction matrix (40 x 4), the trial information delivered post delay, Right: the Pearson correlation between pre-delay and post-delay features. **(b)** SVD of the correlation matrix in A, into a diagonal matrix Σ, and two orthogonal matrices U and V, representing a basis in pre-delay and post-delay features respectively. The bar graph represents the fraction of variance explained by each mode of the decomposition, **(c)** A projection of U values colored by the contingency of the associated trial condition. SVD of the CDL problem naturally divides along contingency into a two-dimensional subspace.

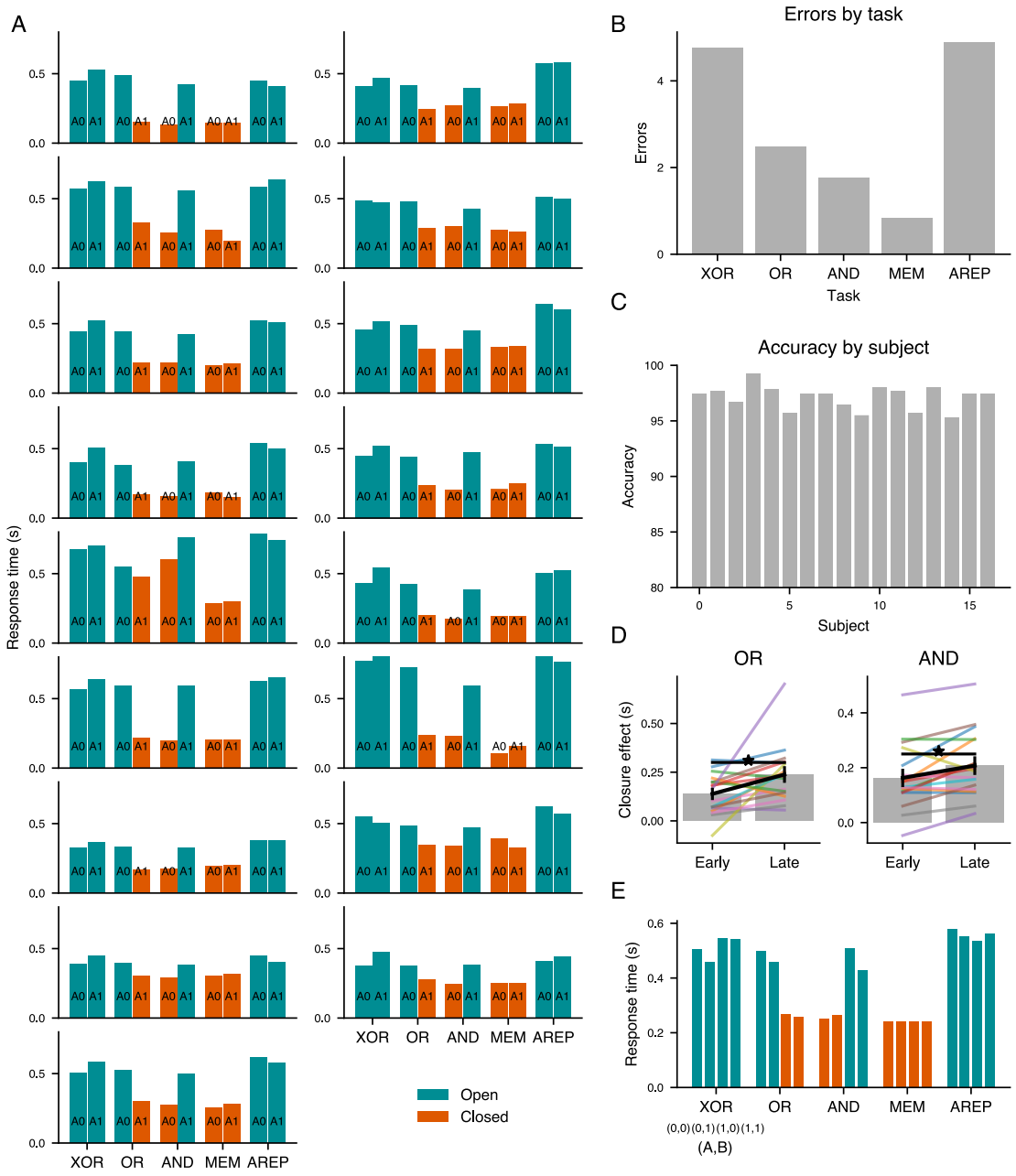


Figure 4.S2: **Individual participant behavior on CDL task.** **(a)** All participant data plotted as in **Fig. 4.2b** with mean reaction time averaged across Cue A and rule pairs. **(b)** Mean errors per task averaged across participants. **(C)** Overall accuracy presented for each participant **(D)** Individual participant data for change in closure effect. Each line represents a single participant, with bars covering the mean early (first two blocks) and late (last two blocks) session closure effect. Black line represents the average change from early to late. **(E)** Mean accuracy by condition, averaged across participants, divided by rule, Cue A and Cue B combination.

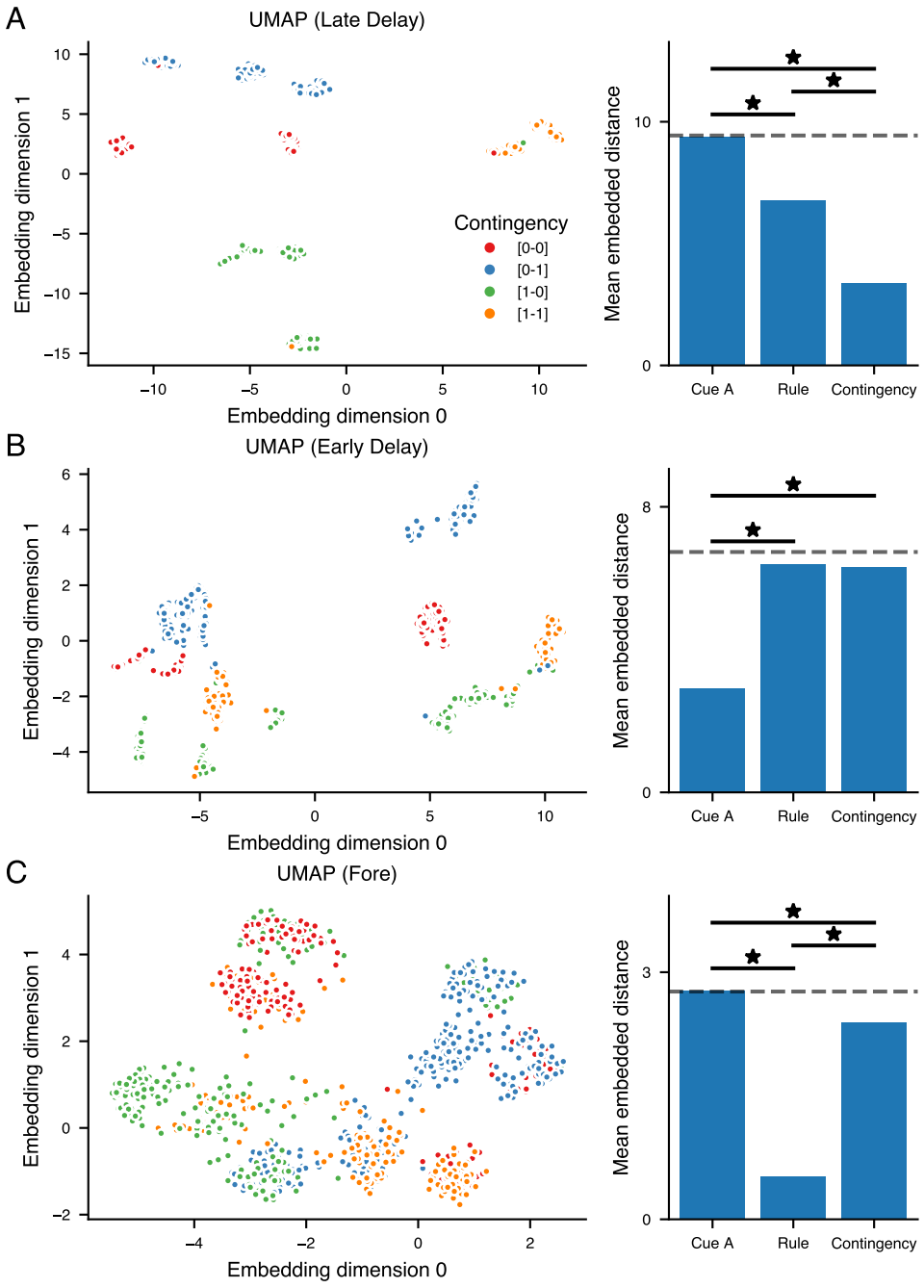


Figure 4.S3: **UMAP analysis of RNN representations.** (Caption on next page.)

Figure 4.S3: (Previous page.) **(a)** Left: UMAP (Uniform Manifold Approximation and Projection) performed on states of the network in the late delay epoch (time point prior to Cue B onset) colored by contingency. Right: Quantification of embedding distance. For each feature first we averaged the embedding values for all trials by condition across that feature (e.g. Cue A=0 and Cue A=1 for Cue A). Then we calculated the mean Euclidean distance from each trial embedding to the its associated centroid. UMAP embeddings in the late delay epoch clustered more substantially by contingency than by Cue A (one-way anova, p¡0.001) or rule (p¡0.001). Dashed line represents chance embedding distance, generated through an analysis of distance to centroids of randomly partitioned trials. **(b)** Analysis as in **(a)** with activity states from the early delay epoch (200 ms after Cue A offset). We found that in the early delay epoch, Cue A centroids significantly outperformed rule (p¡0.001) and contingency (p¡0.001) organization. **(c)** Analysis as in **(a)** with activity states from the fore epoch (50 ms before Cue A onset). In the foreperiod UMAP embeddings were more highly organized by rule than Cue A (p¡0.001) and contingency (p¡0.005). Despite this there remains significant contingency organization during the foreperiod, with lower contingency embedding distance than Cue A (p¡0.001).

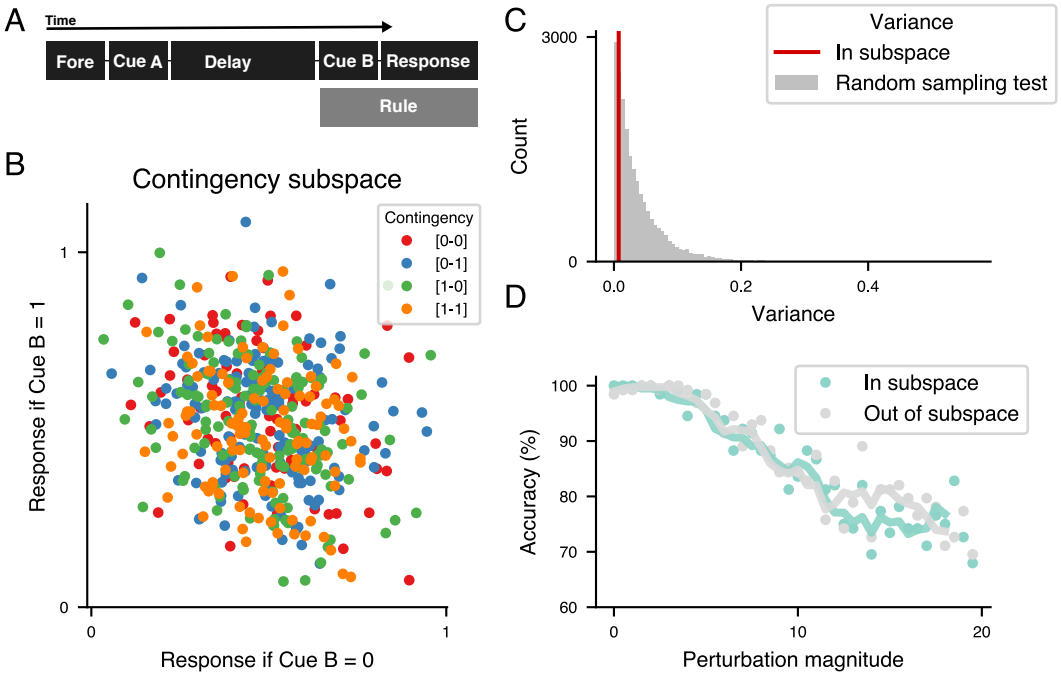


Figure 4.S4: **Late rule task.** **(a)** A schematic of the late rule task. All events in the trial proceed as described for the original task, with the exception that rule input only onsets at the start of Cue B onset rather than being tonically on throughout the trial. This makes calculating contingency during the delay impossible. **b)** The contingency subspace as calculated in **Fig. 4.3b** for the late delay network shows no substantial organization by contingency. **(c)** The contingency subspace in **(b)** shows no preferential variance as compared to randomly chosen subspaces. **(d)** Despite showing no contingency subspace, performance of the network is high with near 100% accuracy in the unperturbed model. Further validating that a contingency solution is not being utilized by this network, the late rule network shows no preferential vulnerability to perturbations in the contingency subspace as opposed to orthogonal to it, for any perturbation magnitude.

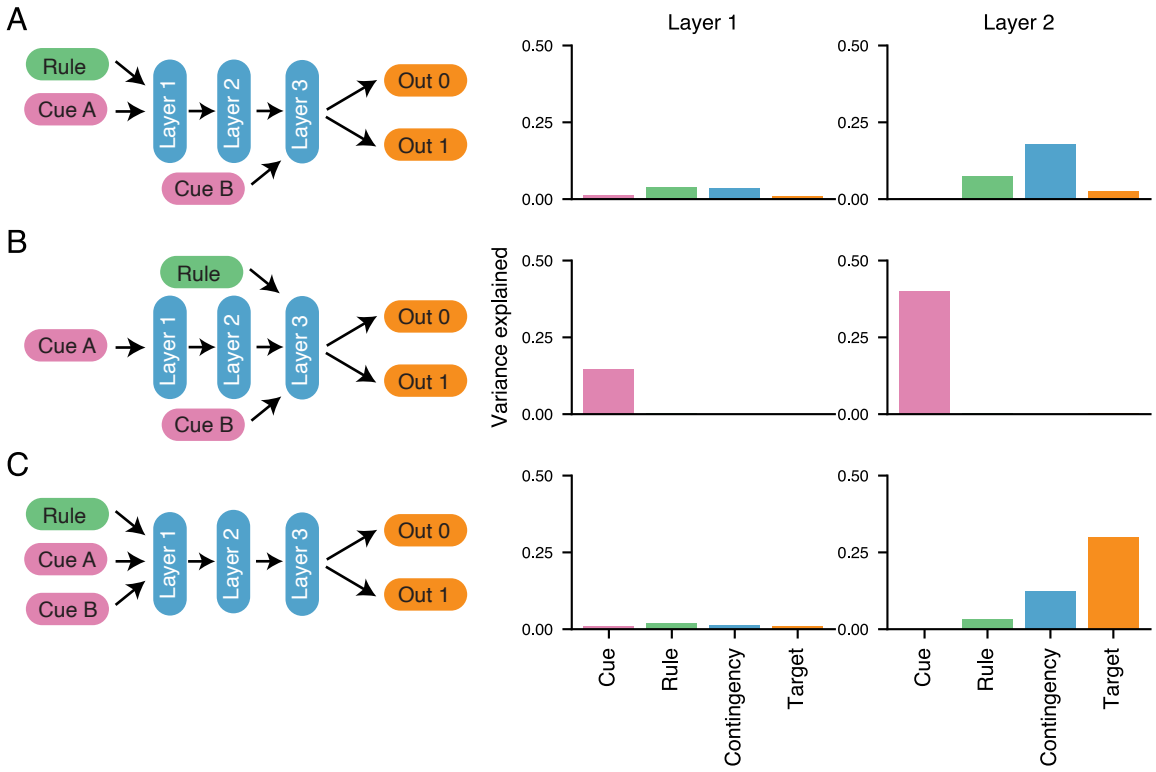


Figure 4.S5: **Feedforward neural networks and stepwise computation.** A feedforward model of the CDL task using a 3 layer fully connected neural network with ReLU non-linearities. Left: Schematic of input regime. Inputs were represented as in the RNN by one hot vectors, with dimensionality 2 for Cue A and Cue B and dimensionality 10 for the rule input. Each hidden layer was composed of 200 ReLUs. All weights were initialized with Glorot initialization. Networks were trained for 20,000 iterations using the Adam optimizer (learning rate=0.001) on a mean squared error loss function with L2 regularization (λ=1) on hidden unit activity. Right: The mean unit state variance explained across trials by each feature (Cue A, rule, contingency and target) for layers 1 and layers 2 of the model. **(a)** As in the CDL task, Cue A and rule are input into the first layer and Cue B is input into the third layer. Information processing of the first layer roughly approximates the early delay, while the second layer represents the late delay and third layer the post delay epochs. Units in the layer 2 (late delay) but not layer 1 (early delay) are best explained by contingency as in the recurrent network. **(b)** As in the late rule network, only Cue A is input into the first layer, with both rule and Cue B being input into layer 3. The network as in the recurrent network demonstrates only Cue A variance in both layers 1 and 2. **(c)** All input, rule, Cue A and Cue B, enters at the first layer. In no layer does contingency best explain unit variance.

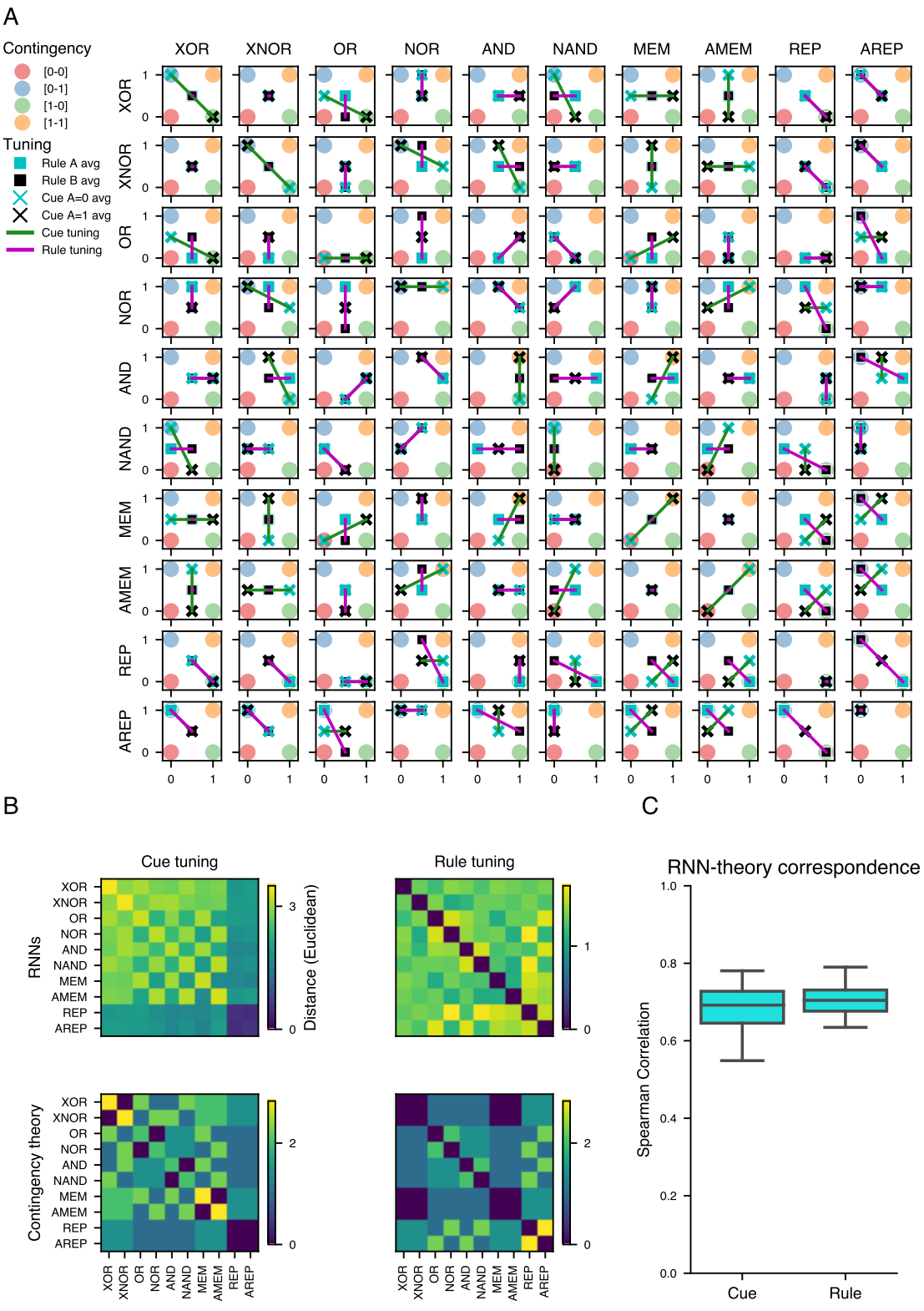


Figure 4.S6: **Task pair tuning analysis.** (Caption on next page.)

Figure 4.S6: (Previous page.) **(a)** Representation for all 55 task pairs of predicted theoretical contingency induced tuning for rule and Cue A. Circles, colored by contingency, represent the centroid for trials of each contingency in the contingency subspace. Cyan(Black) Xs represent the location of trials averaged across rules for Cue A=0(Cue A=1). Cyan(Black) Xs represent the location of trials averaged across rules for Cue A=0(Cue A=1). Cyan(Black) squares represent the location of trials averaged across cues for Rule A(Rule B), with Rule A on the x-axis and Rule B on the y-axis. Green lines represent the direction and magnitude of cue tuning induced by representation in the contingency subspace. Magenta lines represent the magnitude and direction of rule tuning induced by representation in the contingency subspace. **(b)** Measured and theoretical tuning for Cue A and rule. Measured tuning represents the Euclidean distance between the trial averaged recurrent unit states from the late delay epoch. Theoretical tuning is the Euclidean distance between representations averaged, within Cue A or rule for cue tuning and rule tuning respectively, in the contingency subspace. **(c)** Spearman correlation calculated across 20 replicate RNNs between theoretical and measured Cue A and rule tuning. Correlation was measured between lower triangular elements between RNN tuning matrices and contingency theory matrices separately for cue and rule tuning.

Chapter 5

Neural tangent kernel model of human deterministic rule learning

Daniel B. Ehrlich, Sam Zheng, John D. Murray

Abstract

While it is crucial that we learn how to generalize between contexts such that we can apply insight learned in one condition to another, in adverse contexts inappropriate generalization can lead to destructive interference and inhibit learning. Deciding when information should be generalized is therefore a vital concern for learning efficiently, incentivizing the careful construction of priors over what information should be shared. Here we utilize two deterministic logical tasks to investigate across-rule and across-condition learning patterns in human subjects. We use a statistical learning model, a neural tangent kernel (NTK), to decompose the learning process in both tasks into discrete modes. We explore the trade-off between linear modes that utilize low level feature generalization and non-linear modes that can partially correct for failures caused by the limitations of linear feature learning. We find that an intermediate regime of linear and conjunctive representations is capable of explaining learning in both data-sets.

5.1 Introduction

Humans are capable of learning a wide array of behavioral rules to guide their interactions with their environments, from simple stimulus-response reflexes[13, 7] all the way to reasoned plans over abstract concepts[12]. How humans learn rules has been an issue of significant debate. Rule learning can take on different demands depending on the probabilistic or deterministic structure of the environment. In probabilistic environments humans must sample state-action pairs collectively to gain increasingly confident approximations of the optimal rule[23, 4]. In deterministic environments, however, in theory agents should be able to use even just a single instances of each state-action pair to exactly identify rules. Despite this it has long been observed that even in deterministic environments agents rarely can achieve such exact performance and that there are wide variance in the observed alacrity with which different rules over the same stimulus space are learned[22].

The observation that humans do not appear to learn by simply memorizing all the state-action mappings has given rise to a wide set of frameworks regarding how humans learn in such environments[2]. One traditional debate has contrasted prototypes where the agent learns a prototypical model of each category, against exemplar models where the agent stores all examples and matches to the closest one[1]. Further models have instead posited a hierarchical hypothesis testing in which agents first search over simplistic rules and when unsuccessful begin to iteratively search over more complex rules[14]. Lastly groups have suggested that rule learning may be a combination of these phenomena, such as a proposal where predominantly rule learning follows a prototype model but also searches for exceptions[16, 18].

In recent years, the vast increase in the capability of artificial neural network (ANN) models to learn a variety of rules has demonstrated the suitability of less structured learn-

ing processes of acquiring rule respecting behavior[24, 3]. Instead of learning iteratively across different types of representations or using an active hypothesis testing method, ANNs are initialized with large mixed representations and the model, especially as the network increases in width[15], predominantly learns statistically from the structure of the representations available[11, 6].

Here we apply an explicitly statistical learning model, a neural tangent kernel (NTK)[11] approximation to a feedforward ANN, to two deterministic rule learning tasks and observe evidence consistent with human performance over two separate case studies. In contrast to a full feedforward network where we would have to directly simulate weight updates the NTK has two major advantages. First, the reduction of the learning process to a linear dynamical system in the NTK vastly accelerates learning trajectory generation across a variety of conditions. Second, the dependence of the NTK model on a linear kernel enables us to decompose and analyze separable learning modes efficiently and exactly.

In the first study we explore how the order of condition acquisition depends on the overall task structure. As mentioned above, in cases where state-action pairs are memorized they should be independent and only the order of presentation should impact learning, however in alternative learning theories that may not be the case. We constructed a task with two shared rules and a third group-specific rule, to test how inclusion of a rule with different structure could affect learning in the common rules. In the second study we utilized data from the rule learning benchmark task of Shepard and colleagues[22], where they had subjects learn 6 rules over the same stimulus set and observed the differences in average difficulty of acquisition.

Using the NTK model, and analyzing learning contributions from linear feature representation as well as non-linear conjunctions we identify putative explanations for conditional and rule acquisition arising from a purely statistical homogeneous learning process. We identify how weighting higher order feature conjunctions more prominently in

the learning process can impact the order of rule and condition acquisition. Further we demonstrate that by balancing both linear features and conjunctive features correctly we can explain human learning patterns in both tasks in the framework of a simple statistical learning model.

5.2 Results

Study 1: Cross Talk Task

On each trial the participant was first shown a square of one of three colors representing a Rule Cue to be applied during that trial. After 300 ms two binary cues (0 or 1) appeared flanking the rule stimulus. We will refer to the left cue as Cue A, and the right cue as Cue B. The participants could respond either left or right with a button press. After their response they were shown a response screen indicating whether their choice was correct for that trial.

Participants were divided into two groups, with both groups learning two shared rules and each group learning one of two group-specific rules. Participants were not instructed on the correct response for each rule but were tasked to learn the rules from experience and feedback. Both groups learned the XOR and Report B rules. In the XOR rule participants needed to respond right if Cue A and Cue B did not match. In the Report B rule, participants needed to respond right if Cue B was 1 and left if Cue B was 0. In addition to these rules, participants in the AND-group also learned the AND rule, where they needed to respond right if both Cues were equal to 1, and left otherwise. In contrast, participants in the NAND-group learned the NAND rule, where they responded left if both Cues were equal to 1, and right otherwise (Fig 5.1).

Since both NAND and AND are symmetric up to a remapping of the outputs, partici-

	A,B	0,0	0,1	1,0	1,1
Shared Rules	XOR	0	1	1	0
	REPB	0	1	0	1
Group-Specific Rules	AND	0	0	0	1
	NAND	1	1	1	0

Figure 5.1: **AND vs NAND Rule Sets (top)** Shared Rules: The condition-response table for the two rules taught to every participant, XOR and Report B (REPB). **(bottom)** Group-Specific Rules: Condition-response table for the group unique rules. Each participant learned one of the two group-specific rules, AND or NAND, in addition to both shared rules.

pants learning rules independently should show no behavioral effect based on their grouping. If participants are using information across rules to inform their learning, the change in correlation across rules could cause constructive or destructive interference and impact their behavior. We quantify this potential relationship across conditions *imbalance*, defined as the proportion of conditions that share a given feature that lead to the same output (Fig 5.2). Therefore a condition with a positive imbalance in the rule set will benefit from generalization across the feature and learning for a condition with negative imbalance will be impeded.

The same condition could be benefited by condition-wise imbalance across rules in one rule set, but handicapped by imbalance in the other. For example in the case of REP, A=1, B=1, in the AND set it is positively impacted by A-B imbalance as it shares the same output with other conditions with A=1 and B=1 (Fig 5.2, left). In the NAND set however it now is opposed to the majority of the A=1, B=1 conditions giving it a negative imbalance within the rule set (Fig 5.2, right).

Participants were able to learn to perform both tasks to relatively high accuracy (Fig

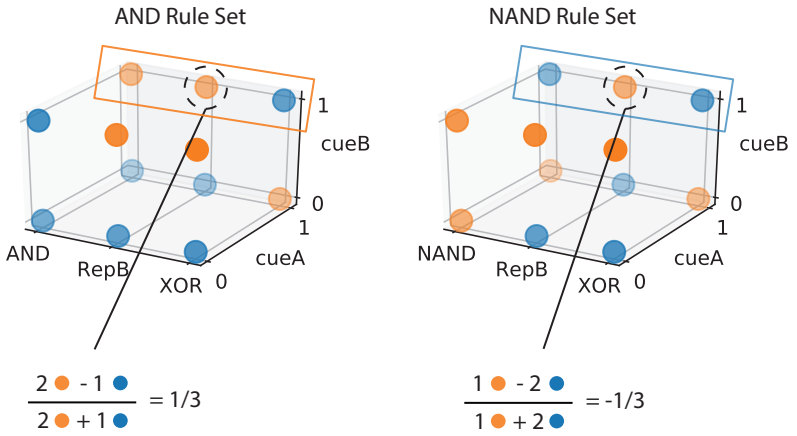


Figure 5.2: **Imbalance Between Rule Sets (left)** Geometric representation of the AND rule set. Each dot represents a specific trial condition (Cue A, Cue B and Rule). Blue dots represent a trial for which the correct response is left, while orange dots represent trials for which the correct response is right. Outlined are all trials for which Cue A = 1 and Cue B = 1. Equations represent the computation for imbalance (here Cue A $\times$ B imbalance). **(right)** Geometric representation for the NAND rule set. All conventions as described for the AND rule set. Notably the equation for imbalance for the highlighted trial changes sign between the sets.

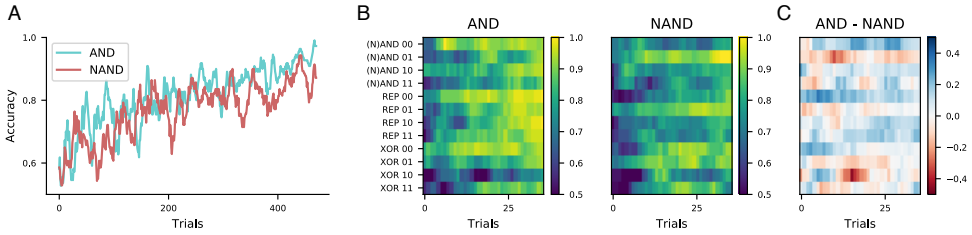


Figure 5.3: **Accuracy Across Conditions and Rule Sets (a)** Mean accuracy across subjects for the AND and NAND rule sets. Accuracy is computed using a 10 trial sliding window approach. **(b)** Mean condition accuracy for each of the twelve conditions in the AND (left) and NAND (right) rule sets. Trials were separated by condition, ordered in terms of presentation and then smoothed using a 5 trial sliding window. **(c)** Difference between average condition accuracy for the NAND and AND rule sets (b).

5.3a). Despite having nearly identical over-all learning trajectories, participants in the NAND group tended to first learn a different set of conditions than did participants in the AND group (Fig 5.3b). To investigate the extent to which imbalance could explain the order of conditions learned, we calculated the correlation between condition accuracy

averaged across subjects and imbalance. We found that while condition accuracy was correlated to Cue A, Cue B and Rule imbalances, it was most highly correlated to the Cue A $\times$ B imbalance, or the imbalance across all other conditions with the same A and B cues in the rule set (Fig 5.S1).

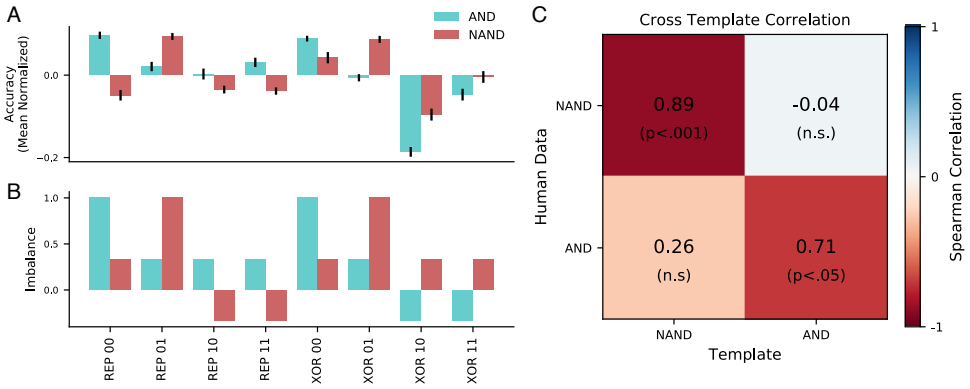


Figure 5.4: **Conditional Accuracy Matches Imbalance (a)** Mean accuracy across time and participants for the AND and NAND rule sets for each of the conditions in the shared rules. **(b)** Imbalance template for AND and NAND rule sets each condition in the shared rules. **(c)** Pearson correlation calculated for the each of the four possible template-behavior comparisons. Significance test performed using permutation test across conditions.

To quantify the effect of Cue A $\times$ B imbalance on learning rate accuracy we formed imbalance templates (Fig 5.4b) that measure the imbalance for each condition in the AND and NAND rule sets. We then used a two proportion z-test to evaluate the proportion of correct trials of a condition across subjects in the AND set verses in the NAND set, deriving a z-score for each condition. The z-score should be negative if for that condition AND is more accurate than NAND, and positive in the opposed case. The z-score matched the direction of the difference between AND and NAND imbalance template for every condition, and was significant for 6 of 8 conditions common to both rule sets (Fig 5.4c).

As a control we wanted to insure that the observed pattern in learning could not be attributed to features general to both rule sets. To do so we calculated the correlation between the condition-wise accuracy averages and the templates, both within a given rule set (e.g. AND template and AND behavior) and across rule sets (e.g. AND template and

NAND behavior). We found that while the correct template explained learning patterns well, the alternative template was a poor match in both directions (Fig 5.3d).

A Neural Tangent Kernel Model Replicates Set Effects

An NTK is an analytical approximation to a neural network of a given structure. It was observed that as a model grows in width, the extent to which each weight changes diminishes. For this reason at an infinitely wide network it can be assumed that the initial weight matrix is approximately fixed. With this assumption in place we can then simulate out the expected learning trajectories of such an infinitely wide ANN model as a linear dynamical system, with learning governed by a between condition kernel[11]. This allows us both to simulate learning trajectories without computing the actual weight updates for an ANN model, but also to directly analyze the structure of learning through the identified kernel.

Given that we utilize the initial weight matrices in the calculation of the NTK, the resulting model is a representation of the inductive biases of the initialized ANN. The basic structure of the population code (e.g. the extent of mixing between features and non linearity in representation of those features) induced by that ANN structure will be passed on to an NTK approximation. As such the structure of the ANN we are approximating using this method will effect both the NTK and thereby the learning dynamics. In this study we used an NTK model to simulate learning for an idealized single layer infinite width ANN model with a rectified linear (ReLU) activation function.

We found that the NTK was able to learn both the AND and NAND task sets and reach high performance across all conditions. As in our human sample(Fig 5.3), there was no difference in mean accuracy trajectories across the two sets (Fig 5.5a). Nevertheless between rule sets, there was substantial deviation between the order in which conditions were learned (Fig 5.5b).

To characterize how well the patterns of learned conditions in our NTK model matched

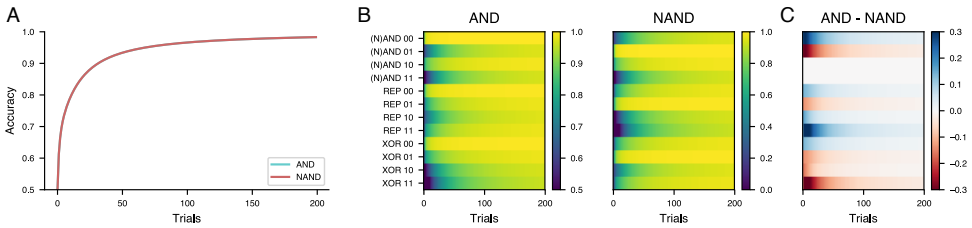


Figure 5.5: **Rule Set Effects in the Neural Tangent Kernel Model (a)** Mean across rule accuracy for the AND and NAND NTK models. **(b)** Across by condition for the (left) AND NTK model and (right) NAND NTK model. **(c)** Difference between the AND and NAND set accuracy for each condition.

the human behavior we took the average accuracy across training for both the NTK and the Human behavior for each condition and calculated the correlation both within rule sets and across rule sets. We found that the NTK trained on the AND task was strongly correlated with the AND human data but not the NAND human data, while the opposite was true for the NTK model trained on the NAND task (Fig 5.6).

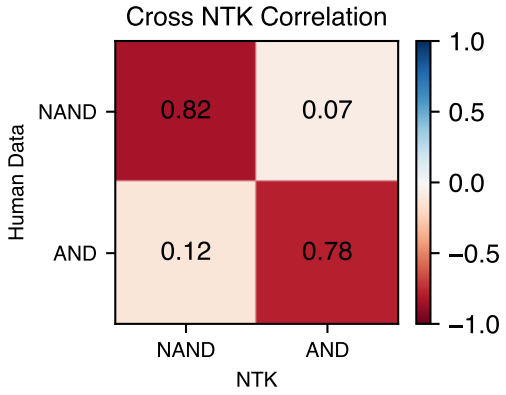


Figure 5.6: **NTK Learning Matches Human Behavior** Pearson correlation calculated between the mean accuracy across human subjects and the mean accuracy across time for the NTK models. Each of the four possible human behavior-NTK models pairs were calculated.

Decomposing Learning Modes From the NTK

The NTK learning dynamics are driven by a kernel which is a linear approximation of the derivative of the output with respect to a change in the weights. We can use an eigen-decomposition of this kernel to identify unique orthogonal modes contributing to learning. This decomposition identified both linear modes that represented the input features, e.g. the Cue and Rule combinations, as well as non-linear modes representing combinations of features 5.7a). In our model we found that the magnitude of non-linear modes tended to be substantially smaller than linear modes 5.7b).

By recombining subsets of the modes into a new kernel 5.7d,f,h) and running the NTK model forward we can analyze the impact of each mode on the order and rate of condition acquisition. Primarily we will be using two subsets of modes, (1) the linear modes which include the constant mode, the rule modes and the cue modes; and (2) the nonlinear modes which include the constant mode and the non-linear conjunctive modes.

We find that while the linear kernel learns more quickly, it cannot asymptotically solve all of the tasks 5.7c). In contrast, the non-linear kernel learns more slowly than the full NTK model but does reach full performance if given sufficient data 5.7e). Perhaps most intriguingly while the linear and full NTK models share the same order of condition acquisition the non-linear model does not indicating the role of linear modes in driving between condition learning rate phenomena.

Study 2: Shepard Task

In brief, each subject was shown a stimulus with binary attributes across three dimensions, size, shape and color. They were tasked with responding either left or right to each stimulus, receiving feedback and learning from experience. The rule applied was randomly drawn from a set of 6 possible rules representing different levels of task complexity (Fig

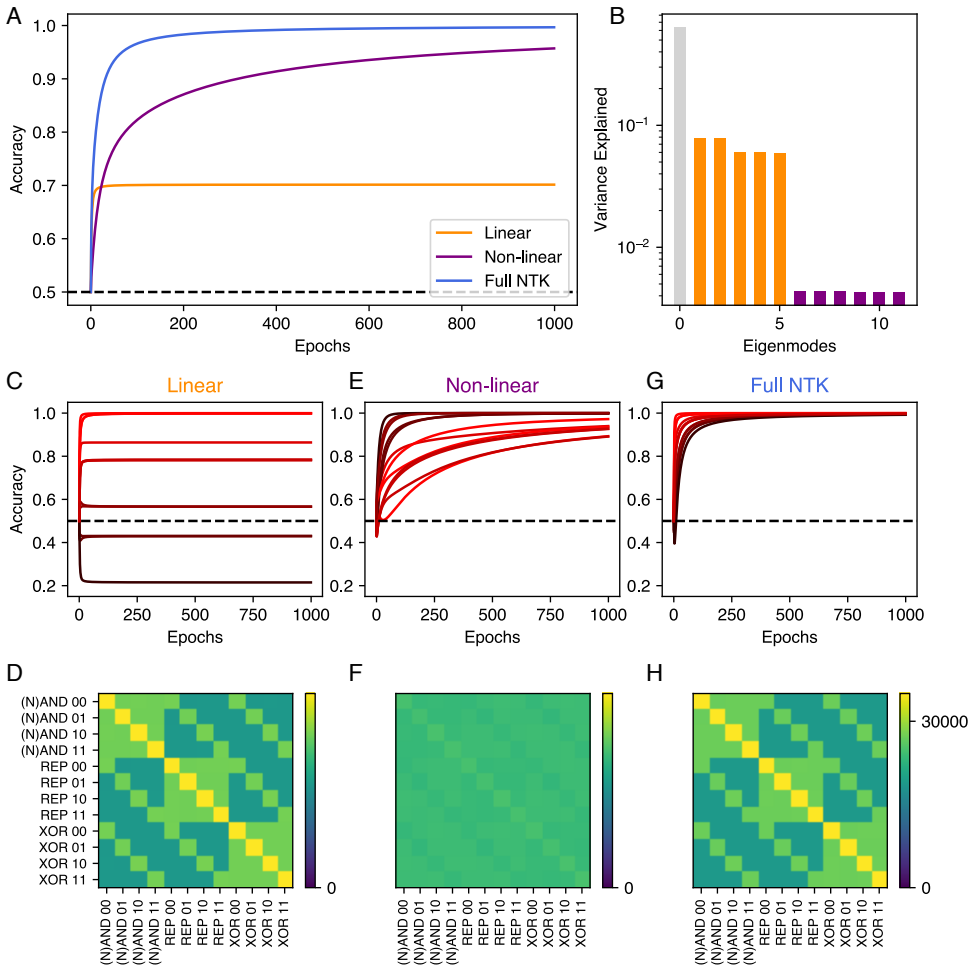


Figure 5.7: **NTK Kernel Decomposition (a)** Accuracy trajectories for partial kernels with only the linear (orange), non-linear (purple) and full kernel (blue) maintained. **(b)** Percent variance explained for each mode of the kernel as well as the mode of across condition variance explained by that kernel. **(c)** Conditional learning trajectories for the linear partial kernel. Color for (c,e,g) indicates asymptotic accuracy for the linear kernel. **(d)** The recomputed partial linear kernel. **(e)** Conditional learning trajectories for the non-linear partial kernel. **(f)** The recomputed partial non-linear kernel. **(g)** Conditional learning trajectories for the full kernel. **(h)** The full kernel.

5.8). Rule I was governed by a single feature, shape. Rule II represented an XOR conjunction of shape and size. Each of Rule III-V were rules governed by a single feature, with one sample of each group swapped. Finally Rule VI was fully conjunctive, having no low level description simpler than the full stimulus-output map itself.

Shepard and colleagues[22], as well as replication studies[17], have demonstrated a

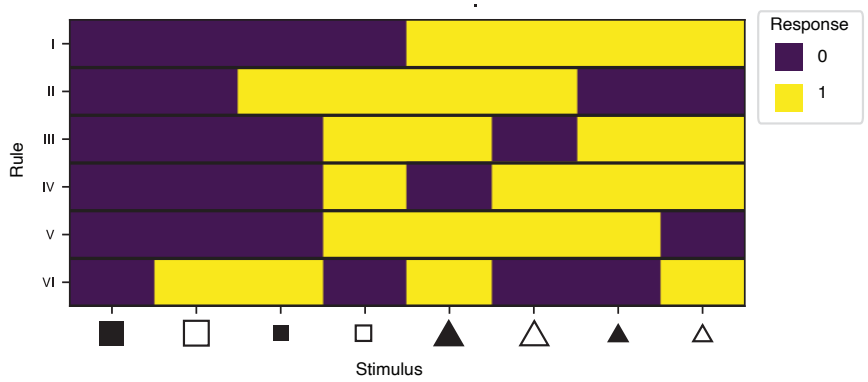


Figure 5.8: **Shepard Task Rules** Representation of the 6 rules utilized in the Shepard task. On each trial the agent saw one of eight possible stimuli with binary variance across three features (shape, size and color). They were tasked with making a binary classification and were shown feedback regarding the accuracy of their choice.

highly consistent order of rule acquisition. Rule I was by far the quickest acquired, followed by rule II, with rules III-V coming next. Rule VI was learned significantly more slowly than were the other rules.

Intriguingly our default NTK model did not closely match this same rule ordering. Notably, in contrast to the human participants rule II was learned second to last only being more quickly acquired than rule IV. Further rules III, IV and V showed significant discrepancy in their learning trajectories. To explore the potential cause of this deviation we decomposed our NTK model into independent learning modes as in the analysis above using a eigendecomposition.

While our default NTK model is a single possible architecture, there is enormous heterogeneity in architectural designs for feedforward networks both at the unit level (e.g. activation function) and at the network level (e.g. number of layers). For this reason we wanted to test the extent to which the form of representations, especially with regard to the composition of linear and non-linear conjunctions, in our one layer feedforward network might explain the deviation between the human and NTK learning results. To do so we

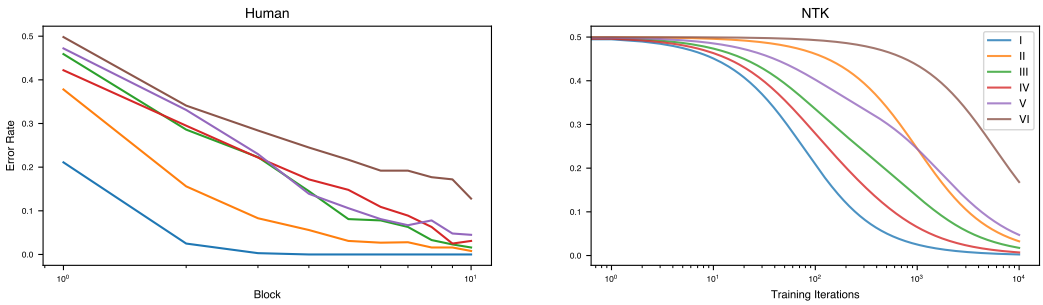


Figure 5.9: **Human vs NTK Rule Learning Trajectories (left)** Human data for the first ten blocks of a replication of the Shepard task. Lines represent average error rate across participants for each of the 6 task rules. **(right)** NTK accuracy data for feedforward network model learning the same task set.

recomputed a new kernel with different ratios of linear and non-linear kernel elements and re-ran the learning trajectory. We found that at a 1:11 ratio we had the best fit to human data ($\rho = .79$). This indicates that humans in the Shephard task are relying on nonlinear conjunctive modes more heavily than does our default NTK model.

5.3 Discussion

Here we use NTK models of a one layer ANN to analyze two phenomena regarding the facility with which humans learn rules. First, in a task well structured to test for the impact of shared feature representations between rules we find that an NTK model of a one layer artificial neural network is a strong match to the order of condition acquisition in human subjects learning the same task. In a decomposition analysis we can link this phenomena to the strong impact of linear feature representation in the learning process. Second, we find the same model does poorly in fitting the order of rule acquisition across rules of different complexity in a traditional benchmark task[22]. However, by modulating the influence of linear and conjunctive modes of learning directly we find an that a network re-balanced to assign greater weight to conjunctive representations fits both condition learning and rule learning well.

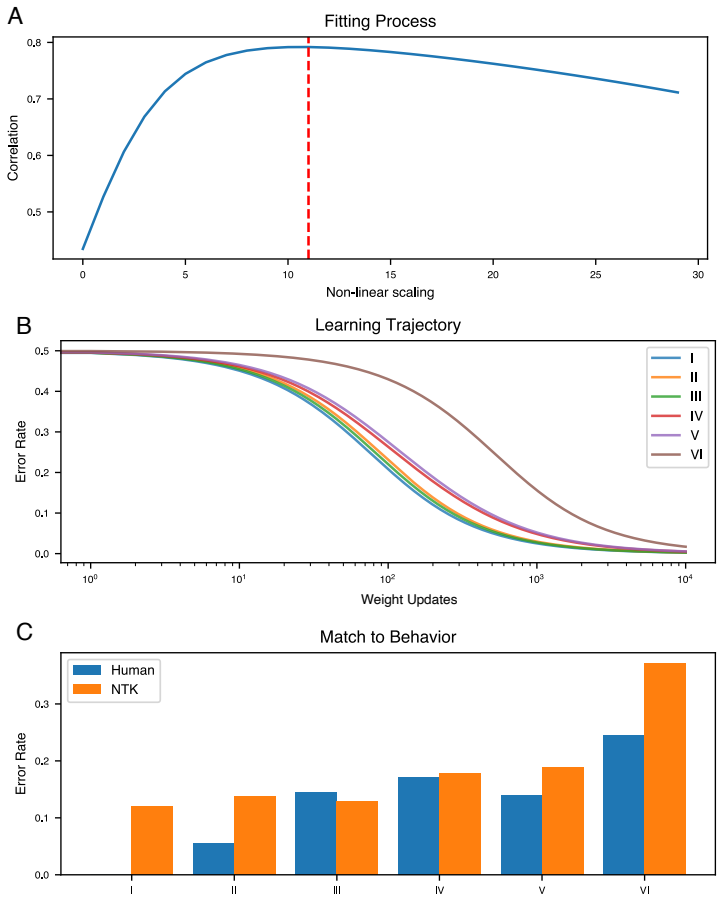


Figure 5.10: **Re-balancing Nonlinearity in Kernel to Match Human Behavior (a)** Values of fit to human behavior measured as across condition correlation in mean accuracy for the human and NTK data. We generated re-balanced kernels, by decomposing the kernel into linear and non-linear learning modes. We then multiplied the non-linear modes by a scaling factor (keeping the linear modes constant), recomposed the kernel and generated new learning trajectories. We found that the best fit was at a scaling factor of 11 (red dashed line). **(b)** The learning trajectories for our best fit model. **(c)** Accuracy for each rule for humans performing the fourth learning block in the Shepard task[17] compared to accuracy for the NTK at the 200th weight update.

In contrast to the NTK approach which fundamentally regards the information as a pattern, learning both linear and conjunctive elements in parallel, several early models of rule learning utilized active hypothesis testing to check for the feasibility of simple rules that align with a small number of features. For example the RULEX model attempted to iteratively search over one dimensional rules, then simple rules with minimal exceptions,

followed only last by full pattern recognition[18]. Other related approaches sought to use low level category representations, only forming more complex conjunctive representations if met with low accuracy rates[14]. In contrast to these approaches we used a model that learned from all levels of representation simultaneously. Intriguingly we found that the seemingly ordered behavior could be well explained by weighting different complexity representations appropriately.

One commonly observed issue in deep learning is the data inefficiency of many current state of the art methods in deep learning[10]. In contrast to human learners who can acquire tasks with a small number of representative samples, deep learning models often require extremely large datasets with enormous numbers of iterations to reach equal performance[21]. Here we observe an example of efficient data use in both our human and NTK learning. While stimulus information across rules is neither necessary nor sufficient to solve the task, relying on only each trial condition independently requires many more samples to reach high performance than to share information across conditions[20, 5].

This increase in data efficiency through implicit data augmentation[20] is a core advantage of multi-task learning, or learning a task simultaneously with an auxiliary task. One potential issue however is that where tasks diverge the use of data from one task may cause artifacts in the other task. This divergence can be thought of more broadly, but in our cross-talk task maps neatly to the idea of imbalance potentially explaining why imbalance templates can explain learning patterns so closely.

As a deterministic task, one strategy an agent could in theory use is to simply memorize each mapping. In this case an agent should reach 100% performance following the first twelve trials. Despite this, we don't find this behavior in either our human or NTK model. In the NTK model, learning is by definition incremental so it is unsurprising, but less so in the human data. There are several reason humans may not adopt this strategy. First, we did not explicitly tell our participants the task is deterministic or that conditions

exclusively mapped to one output. They may decide to sample either output for the same condition to determine the task meta-structure themselves. Second, it may not be possible for participants to separately memorize each condition directly[8]. A limit on working memory could force subjects into a less deliberative, and more gradual, learning regime. Third, subjects may attempt to directly memorize but their memory may be corrupted by trials with shared features[19]. A trial could potentially be acting as a "distractor" for future trials leading to the incorrect choice.

An early issue for artificial neural networks was the inability for early systems to learn certain non-linear problems (e.g. the XOR problem)[9]. These issues have been solved with the rise to popularity of highly non-linear deep networks capable of dissociating complex representations. In this study we find that learning in both humans and a relatively simple, but potentially highly non-linear model, were surprisingly linear. Instead of non-linearity providing the dominant role in learning, it played an accessory role, working to counterbalance the directions in which linearity may inhibit learning, and the comparison to human learning patterns began to break down as non-linearity began to dominate.

In conclusion, we observed participants learning deterministic rules integrated information both linearly across conditions as well as through higher order nonlinear conjunctions. We observed a close correspondence to the order of condition acquisition seen in human participants using an NTK model of a single layer feedforward network. By decomposing the modes involved in learning in the NTK model, we isolated a regime of balance between linear and nonlinear learning components that could explain differences in learning rate between rules requiring different nonlinear complexity. This research indicates the sufficiency of a largely unstructured learning processes, assuming the correct inductive biases over information shared across conditions, to explain human learning patterns.

5.4 Methods

Imbalance

Imbalance for a condition was defined as:

$$\mathbf{I}_i = \frac{\sum_j^{N_f} m_{ij}}{N_f} \tag{5.1}$$

$$m_{ij} = \begin{cases} 1 & \text{if } f_i = f_j \\ -1 & \text{if } f_i \neq f_j \end{cases} \tag{5.2}$$

where I_i is the imbalance for condition i, N_f is the number of conditions that share the same feature as condition i, m_{ij} is an indicator variable to indicate whether condition i and j share the same response. f indexes for the feature imbalance is being calculated across.

Task Descriptions

Cross Talk Task

Participants were recruited on the MTurk platform. Task structure was explained and participants were shown an example stimulus.

We recruited a total of 50 participants (ages: 21-69, 44% male). All participants were shown and accepted an online informed consent document approved by the Yale Institutional Review Board (IRB) and were paid for their participation.

Participants were excluded from analysis if their overall accuracy across rules was less than 65%. 16 of 31 total participants were excluded from the NAND group, while 7 of

19 total were excluded from the AND group, leaving us with 15 NAND and 12 AND participants in the final analysis.

Participants maintained anonymity and were paid through the MTurk platform.

Following training participants were shown 480 trials in a pseudo randomized order (40 of each Rule x Cue A x Cue B condition) broken up into blocks. On each trial the participant were first shown a square of one of three colors representing a Rule Cue to be applied during that trial. After 300 ms two binary cues (0 or 1) appeared flanking the rule stimulus. We will refer to the left cue as Cue A, and the right cue as Cue B. The participants could respond either left or right with a button press. After their response they were shown a feedback screen for 800ms indicating whether their choice was correct for that trial.

Participants were divided into two groups, with both groups learning two shared rules, and each group learning one of two group-specific rules. Both groups learned the XOR and Report B rules. In the XOR rule participants needed to respond right if Cue A and Cue B did not match. In the Report B rule, participants needed to respond right if Cue B was 1 and left if Cue B was 0. In addition to these rules, participants in the AND-group also learned the AND rule, where they needed to respond right if both Cues were equal to 1, and left otherwise. In contrast, participants in the NAND-group learned the NAND rule, where they responded left if both Cues were equal to 1, and right otherwise.

We collected 11 participants for the AND set and 14 participants for the NAND set.

Shepard Task

For this task we used the first ten blocks of learning data from the implementation in the replication study performed by Nosofsky and colleagues[17]. On each trial a participant was shown a stimuli that varied along 3 dimensions (shape, size and color) and registered their response with a key press. For each rule, each participant was first shown two blocks

each of 8 trials where each stimulus was shown once in random order. The following blocks were each of 16 trials, with each stimuli occurring twice. After each trial, the participant was instructed as to whether they had made the correct choice.

Each participant learned two rules. Participants were explicitly instructed that the rules were independent and the order and rule pairs were counterbalanced across participants.

Behavioral Analysis

Accuracy across time was evaluated on a moving window basis of 10 trials (Fig 5.3a).

Condition specific accuracy was calculated based on the number of trials of that condition the participant had already seen, (e.g the third time the participant saw XOR, A=0, B=1 was the third trial of that condition). Accuracy per condition was calculated with a 5 trial smoothing window (Fig 5.3b).

For each rule set, we first calculated the AB imbalance for each condition forming an imbalance template. We averaged accuracy across time to get a single scalar value for accuracy through the task for each condition. We then used Spearman correlation to measure the match between accuracy and imbalance. We used a leave one subject out method to estimate error bars.

To measure the rule set effect within each condition we measured the number of correct and total choices for each condition and each group. We then used a two-proportion z-test to measure the direction, magnitude and significance of the effect. We found that all difference in accuracy between rule sets matched the direction of the imbalance difference between rule sets with six of eight reaching significance (p¡.05).

NTK Model

Our NTK model approximates an infinite width 1 layer feedforward ANN with ReLU activation functions. The feedforward pass is governed by the equations:

$$\mathbf{H} = f(\mathbf{X}\mathbf{W}_{in}) \tag{5.3}$$

$$\mathbf{o} = \mathbf{H}\mathbf{W}_{out} \tag{5.4}$$

$$\mathbf{p} = sigmoid(\mathbf{o}) \tag{5.5}$$

$$L = -\sum_{i=1}^{N_{conditions}} y^i log p^i + (1 - y^i) log(1 - p^i) \tag{5.6}$$

X and y are the network input and output respectively. W_in are the weights from the input to the hidden layer. W_out are the weights from the hidden layer to the output. o is the output of the network, and p is the output passed through a sigmoidal nonlinearity. H is the hidden layer activity.

Traditional neural network training utilizes back-propagation of error gradients to instruct weight updates, here we use the same update equations to derive the expected update for the output predictions p.

$$\nabla_{\mathbf{p}} L = -(\frac{\mathbf{y}}{\mathbf{p}} - \frac{1-\mathbf{y}}{1-\mathbf{p}}) \tag{5.7}$$

$$\nabla_{\mathbf{w}} \mathbf{p}(\mathbf{W}) = \nabla_{\mathbf{w}} \mathbf{o}(\mathbf{W}) \cdot \nabla_{\mathbf{o}} \mathbf{p}(\mathbf{o}) \tag{5.8}$$

$$= \nabla_{\mathbf{w}} \mathbf{o}(\mathbf{W}) \cdot diag(\mathbf{p} - \mathbf{p}^2) \tag{5.9}$$

$$\dot{\mathbf{W}} = -\nabla_{\mathbf{w}} L(\mathbf{W}) \tag{5.10}$$

$$= -\nabla_{\mathbf{w}} \mathbf{p}(\mathbf{W}) \cdot \nabla_{\mathbf{p}} L \tag{5.11}$$

$$= -\nabla_{\mathbf{w}} \mathbf{o}(\mathbf{W}) \cdot (\mathbf{p} - \mathbf{y}) \tag{5.12}$$

To approximate $\nabla_{\mathbf{w}} \mathbf{o}(\mathbf{W})$ we concatenate all W_in and W_out into a 1D vector. Specifically:

$$\frac{\partial o[n]}{\partial W_{in}[i,j]} = x[n,i] \times (H[n,j] > 0) \times W_{out}[j] \tag{5.13}$$

$$\frac{\partial o[n]}{\partial W_{out}[j]} = H[n,j] \tag{5.14}$$

This can be considered a feature map: $\phi(\mathbf{X}) = \nabla_{\mathbf{w}} \mathbf{o}(\mathbf{W})$, which will be used to define the kernel: $\mathbf{K} = \phi^T \phi$. The kernel is parameterized by $\mathbf{W}$, which changes during training.

The goal is to define equations that allow us to approximate the update to p directly without having to update the weights of our network. By utilizing the fact that weight update magnitudes decrease as the size of the network increases we can use the initial values of W to approximate the kernel throughout training.

$$\dot{\mathbf{p}(\mathbf{W})} = \nabla_{\mathbf{w}}\mathbf{p}(\mathbf{W})^T \cdot \dot{\mathbf{W}} \tag{5.15}$$

$$= -diag(\mathbf{p} - \mathbf{p}^2) \cdot \nabla_{\mathbf{w}}\mathbf{o}^T(\mathbf{W})\nabla_{\mathbf{w}}\mathbf{o}^T(\mathbf{W}) \cdot (\mathbf{p} - \mathbf{y}) \tag{5.16}$$

$$= -diag(\mathbf{p} - \mathbf{p}^2) \cdot \mathbf{K}(\mathbf{W}) \cdot (\mathbf{p} - \mathbf{y}) \tag{5.17}$$

$$\approx -diag(\mathbf{p} - \mathbf{p}^2) \cdot \mathbf{K}(\mathbf{W}_0) \cdot (\mathbf{p} - \mathbf{y}) \tag{5.18}$$

NTK Simulation

Cross-Talk task: input was presented with the rule and stimuli indicated by separate one-hot encodings. Output of the NTK was determined as a one-hot encoding determined by the rule-set being simulated. We simulated the networks forward using Euler integration for 1000 trials with a learning rate of 0.01 for each of the two rule sets.

Shepard task: input was presented with one-hot encodings separated by feature(shape, color and size). Output of the NTK was determined as a one-hot encoding determined by the rule being simulated. We simulated the networks forward using Euler integration for 10000 trials with a learning rate of 0.001, for each of the 6 rules.

NTK Analysis

We decomposed the NTK kernel (Fig. 5.9) using an eigen-decomposition into orthonormal condition modes and associated scalar eigenvalues.

$$K(\mathbf{W}_0) = Q\Lambda Q^{-1} \tag{5.19}$$

We generated lesioned kernels by recombining a subset of modes from the eigen-

decomposition. For the linear kernel, we used just the first 6 modes of our decomposition. For the non-linear kernel we used the first mode, constant mode of the kernel, as well as the non-linear modes 7-12.

$$K(\mathbf{W}_0)_{lin} = Q_{:6}\Lambda_{:6}Q_{:6}^{-1} \tag{5.20}$$

$$K(\mathbf{W}_0)_{nl} = Q_{1,7:}\Lambda_{1,7:}Q_{1,7:}^{-1} \tag{5.21}$$

We simulated the NTK as above, with the exception of the use of a lesioned kernel, instead of the full NTK kernel.

For the linear-nonlinear analysis (Fig. 5.10), we followed a similar procedure except instead of recalculating a kernel using only a subset of modes we recalculated it with non-linear modes all multiplied by a scalar factor. We searched over factors of different magnitudes. We simulated out the learning process with these re-weighted kernels and calculated the correlation between the human and NTK across rule accuracy, with the weights with greatest correlation being determined as the best fit.

Bibliography

[1] F Gregory Ashby and W Todd Maddox. Relations between prototype, exemplar, and decision bound models of categorization. *Journal of Mathematical Psychology*, 37(3):372–400, 1993. 95

[2] F Gregory Ashby and W Todd Maddox. Human category learning. *Annu Rev Psychol*, 56:149–78, 2005. 95

[3] Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Curr Opin Neurobiol*, 46:1–6, 10 2017. 96

[4] Timothy EJ Behrens, Mark W Woolrich, Mark E Walton, and Matthew FS Rushworth. Learning the value of information in an uncertain world. *Nature neuroscience*, 10(9):1214, 2007. 95

[5] Rich Caruna. Multitask learning. *Machine Learning*, 28:41–75, 1997. 108

[6] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *arXiv*, 2018. 96

[7] Anne G E Collins and Michael J Frank. Cognitive control over learning: creating, clustering, and generalizing task-set structure. *Psychol Rev*, 120(1):190–229, Jan 2013. 95

[8] Randall W Engle. Working memory capacity as executive attention. *Current Directions in Psychological Science*, 2002. 109

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. 109

[10] Hlynur Davio Hlynsson, Alebrto N Escalante-B., and Wiskott Laurenz. Measuring the data efficiency of deep learning methods. *arXiv*, 2019. 108

[11] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in Neural Information Processing Systems*, 31, 2018. 96, 101

[12] Andrew S Kayser and Mark D'Esposito. Abstract rule learning: the differential effects of lesions in frontal cortex. *Cereb Cortex*, 23(1):230–40, Jan 2013. 95

[13] Atsushi Kikumoto and Ulrich Mayr. Conjunctive representations that integrate stimuli, responses, and rules are critical for action selection. *Proc Natl Acad Sci U S A*, 117(19):10603–10608, 05 2020. 95

[14] Bradley C Love, Douglas L Medin, and Todd M Gureckis. Sustain: a network model of category learning. *Psychol Rev*, 111(2):309–32, Apr 2004. 95, 108

[15] Alexander G. de G. Matthews, Mark Rowland, Jiri Hron, Richard E. Turner, and Zoubin Gharamani. Gaussian process behaviour in wide deep neural networks. *arXiv*, 2018. 96

[16] Daniel J. Navarro. Analyzing the rulex model of category learning. *Journal of Mathematical Psychology*, 49(4):259–275, 2005. 95

[17] R M Nosofsky, M A Gluck, T J Palmeri, S C McKinley, and P Glauthier. Comparing

models of rule-based classification learning: a replication and extension of shepard, hovland, and jenkins (1961). *Mem Cognit*, 22(3):352–69, May 1994. 104, 107, 111

[18] Robert M Nosofsky and Thomas J Palmeri. A rule-plus-exception model for classifying objects in continuous-dimension spaces. *Psychonomic Bulletin & Review*, 5:345–369, 1998. 95, 108

[19] Xue-Lian Qi, Anthony C Elworthy, Bryce C Lambert, and Christos Constantinidis. Representation of remembered stimuli and task information in the monkey dorsolateral prefrontal and posterior parietal cortex. *J Neurophysiol*, 113(1):44–57, Jan 2015. 109

[20] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv*, 2017. 108

[21] Thomas Serre. Deep learning: The good, the bad, and the ugly. *Annu Rev Vis Sci*, 5:399–426, 09 2019. 108

[22] Roger N Shepard, Carl I Hovland, and Herbert M Jenkins. Learning and memorization of classifications. *Psychological Monographs: General and Applied*, 75(13):1–42, 1961. 95, 96, 104, 106

[23] Leo P Sugrue, Greg S Corrado, and William T Newsome. Matching behavior and the representation of value in the parietal cortex. *Science*, 304(5678):1782–7, Jun 2004. 95

[24] Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nat Neurosci*, 22(2):297–306, 02 2019. 96

5.5 Appendix

Supplementary Figures

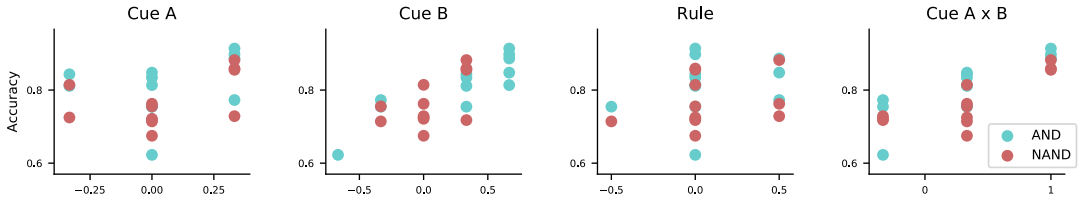


Figure 5.S1: **Imbalances by Accuracy for Each Task Feature** We calculated the imbalance for each feature (Cue A, Cue B, Rule and Cue A $\times$ B interaction). We then took the average across subject accuracy for each condition for the AND and NAND set separately (plotted in blue and orange respectively).

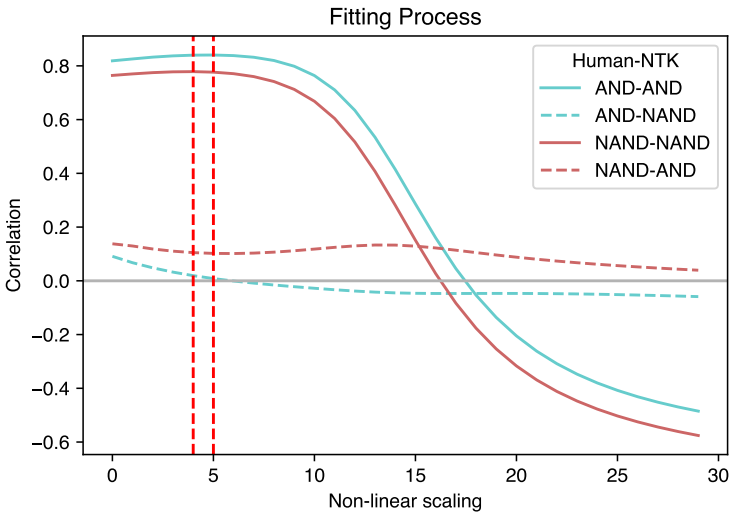


Figure 5.S2: **Nonlinear Scaling in the Cross Talk Task** Values of fit to human behavior in the Cross-Talk task measured as across condition correlation in mean accuracy for the human and NTK data. We generated re-balanced kernels, by decomposing the kernel into linear and non-linear learning modes. We then multiplied the non-linear modes by a scaling factor (keeping the linear modes constant), recomposed the kernel and generated new learning trajectories. We separately compared for each scale factor the AND/NAND NTK to both the corresponding and alternative rule set.

Chapter 6

Discussion

6.1 Conclusions and Perspectives

The three discrete research projects included in this dissertation, each complete in answering a specific set of targeted questions, collectively add new insight more broadly regarding basic cognitive processes as well as methodological considerations for work in ANN models.

As mentioned above, one common theme between projects presented in this dissertation is the specific challenges and demands regarding dealing with learning and implementing multiple rules in conjunction. In Chapter 3 we discuss our RNN training software PsychRNN which was specifically built to enable the expression of complex multifaceted task environments. In Chapter 4 we explicitly use a task that is built off of ten possible rules, examining the ways in which rules can be combined and re-used to more efficiently solve them. Finally in Chapter 5 we explore how interference between rules impacts learning and thereby behavior.

The emergence of either constructive and destructive interference in human behavior, as was observed in the CDL and Cross-Talk tasks, acts in many ways as a a constraint on the possible algorithms that are being utilized[39]. In the CDL task we see evidence of

”representation bias”[33] whereby the inclusion of multiple tasks incentivizes the agent to identify representations efficient across the presented tasks. In the Cross-Talk task in contrast we see evidence of ”implicit data augmentation”[33], or the ability of an agent to use information from one task to at least partially learn a different task. In the case of the study presented we can see how such a strategy while aiding in increasing the data efficiency of the task (and thereby decreasing training time) imparts biases due to the partial overlap between tasks.

While the models used to describe multi-rule task learning in Chapter 5 do not have precise and specific task structure, the ”meta-structure” or proportions of different levels of conjunction and non-linearity are both free parameters in the design of system and relevant to how well the models fit to human behavior.

An issue of recent interest to the field of systems neuroscience, and relevant to the work presented in this dissertation, is skepticism regarding the utility of trying to map neural states to ”representations”[3]. While early sensory neuroscientists were able to directly link the firing rates of neurons to external states [17, 27, 19], more recently the firing rates of neurons have been less easily tied to a comprehensible unified and externally definable value[31]. In addition, observations in ANN models have shown that network without units assigned to specific values can do many cognitive tasks of interest[38]. While we utilize the language of representations in our explanation of units being variant in their activity as a function of contingency on that trial, it would be equally valid to neglect representation entirely and frame that variance through the lens of computational bottlenecks. While this debate is in many ways primarily semantic, the inclusion of computational subroutines, even where they are substantially more abstract than contingency, as a possible organizing pattern behind unit variance may provide more insight than trying to explain neural firing through specific external variables.

One important question as neuroscience as a field matures in its use of ANN models

is how to design tasks that can provide the most insight[4, 30]. Currently tasks tend to be designed with respect to human, primate or rodent equivalents[24, 26]. This strategy, based on the goal of eventually comparing results between the model and collected data is a reasonable starting point, but it neglects many of the greatest strengths of ANN models. Tasks trained in actual human or non-human agents tend to be constrained to a relative simplicity as in human experiments you rarely have more than a few hours of sessions to not only fully teach the task but also to collect data. In animals, the cognitive capacity of the model organism being used acts as an upper-bound on task complexity.

ANN models however need not be bound to few task conditions and can learn a large number of simultaneous tasks or conditions[39]. Including greater variety of trials in a single task greatly facilitates reverse engineering algorithmic level solutions from trained ANN models. Meanwhile, by embedding tasks from human or animal experiments as ”sub-components” of the overall trained task you can still use the model as a comparison to collected experimental data.

As important as choosing the right task is utilizing analyses methods well suited to ANN models. While there are many examples of papers using custom-designed analysis to reverse engineer their networks[29, 24], and there are a few attempts to generalize solutions to the problem of going from trained network to algorithm[13, 37] it is hard to design analyses that will be useful in all studies. Instead, it may be more productive to think of types of analyses that have special utility or feasibility in ANN models. Notably methods that benefit from precision, full observability and between task comparisons can be implemented much more readily in ANN models than in humans or animal models.

With the popularization of optogenetics[8] and its application in a variety of model organisms[23, 14] has demonstrated the potential utility of perturbation experiments to unraveling circuit function. Several limitations to such methods in animal models are diminished or removed entirely in ANN models. In ANN models perturbation can be

exact in their targets only impacting specific sets of units, they can be precisely reproduced multiple times, and they can be restricted to arbitrarily precise time points. Further the removal of total trial number limitations means that this precision can be used to great effect in testing the temporal role of different potentially overlapping subsets of units.

This property of near unlimited trial numbers further assists in out of sample task testing, or exploring behavior and circuit behavior in an environment different from the one the agent was trained on. This type of analysis has several problems in human and animal research. First, humans and animals continually learn so after the emergence of a new task they begin to adapt immediately making only the first few trials truly reflective of the baseline strategy[11]. In ANN models the weights can be fixed after training, so in a novel environment you can be free to run as many trials as you need to get sufficient analytical power. Second, human and animal agents are time limited in their ability to continue performing variants of a task forcing a trade-off between power on one task verses multiple tasks. Due to the effectively unlimited trial numbers for modeling generally ANNs can be put through dozens of out of sample tests to precisely define the task features required to allow the network to perform accurately.

6.2 Future Directions

While the work presented above provides insight and clarity with regards to long present questions in the fields of prefrontal cortical neurobiology, working memory and rule learning, there remain many open areas of research which can benefit from additional investigation.

Computational psychiatry uses theory and tools taken from computational neuroscience and applies it to understanding psychiatric illness[1, 21]. As many psychiatric diseases have heterogeneous symptoms that rarely are constrained to a single cognitive process[35],

our work understanding how different processes may be implemented conjointly could provide insight as to the source of neurocognitive dysfunction.

Common tasks found to be deferentially impacted in patients with schizophrenia sit at the nexus of cognitive control, memory and decision making[18]. Traditionally these deficits have been interpreted as a working memory failure[34], however in recent years a primate model has raised questions about the neurobiological source of deficits[6, 5]. Through using the methods described above, training an RNN model to perform tasks of clinical interest in schizophrenia research[10] we can gain insight into the possible mechanisms that subserve behavior in healthy models. Then by implementing perturbations to the trained network in ways that mimic observed neural phenomena in schizophrenia we may get insight into the failure modes that lead to disease symptomology. Specifically impacts to excitatory-inhibitory tone, which is strongly altered in patients with schizophrenia[20, 36], we can identify possible connections between cellular level deficits and their resulting impact on circuit, algorithmic and cognitive processing.

The work presented here predominantly uses basic and flexible artificial neural network models. In human and animal neural circuits there exists substantial constraints on the connections within and between regions. As we demonstrate in chapter 3, the PsychRNN toolbox is capable of implementing RNNs with a variety of connectivity constraints, for example Dale's law and sparse connectivity. Using these methods we can analyze how distributed networks of multiple regional sub-networks with different input-output connectivity may develop separate functional specializations as well as coordinate to facilitate cognitive computation.

While in this work we focus specifically on computations believed to be subserved by neuronal populations in the PFC, abstracting away activity in other areas of the brain[12], in reality the PFC works as a component of multiple functional networks distributed across cortical and subcortical areas [25, 2, 7]. While there are likely many other differentiating

factors between these areas[28], one key difference is their organization with respect to primary sensory cortices. By generating a modular model with each region receiving different input modalities it is possible to simultaneously investigate the mechanisms underlying regional specialization in working memory as well as possibly explain the coordination between regions.

As specific extension of the work presented in this thesis, circuits in the thalamus have been identified to help coordinate task related activity in PFC during multi-rule tasks [15, 32]. In addition to this specific structural specialization of the thalamus and reticular nucleus provide intriguing constraints on their computational role [22, 16]. Using the connectivity parameters implemented in PsychRNN it would be possible to implement a circuit matching the known neurobiological structure of PFC-thalamic loops, and investigate the emergent rise of functional specialization as well as inter-areal coordination in a trained model.

Lastly while the PFC has long been a locus for research regarding working memory, complementary research in other regions of cortex have identified persistent task related neural states. Intriguingly, while PFC seems to most generally be recruited during many types of working memory tasks other regions appear to be more domain specific[9].

In a related direction, we can investigate how unit-specialization constrains different solutions. Enforcing Dale's law, or constraining each unit to have exclusively excitatory or inhibitory post-synaptic connections, can provide insight into how neurons with excitatory and inhibitory primary neurotransmitters may be contributing to brain function.

One area of tension that remains from our CDL and NTK projects is the extent to which we believe representations in the prefrontal cortex as well as elsewhere in the brain can be conceptualized as largely unstructured. In the CDL project we identify a specific and highly structured underlying representation. In contrast, for the NTK project we specifically use a model incapable of fundamentally transforming the logic of the initial

representation, leaving learning dynamics as a function of the initial unstructured connectivity.

A possible explanation for why we see human behavior as consistent with a structured model in one task and consistent with an unstructured model in another is that humans are capable of using multiple systems in different contexts. For this reason a potentially useful direction would be to explore experimental designs that share different features with either the CDL task or the NTK task in terms of design or training. For example we can explore how different types of training (instructed vs experiential), different types of stimuli (verbal or shapes) or different types of designs (delay or immediate) may impact the adoption of structured or unstructured representational strategies.

Lastly, while this work explores two novel behavioral paradigms using modeling to infer possible neurocomputational underpinnings it is important to directly investigate the neural signals during task performance to disqualify alternative possible mechanisms. For the CDL task we make specific and thorough predictions of both the signal and the structure of neural representations consistent with contingency coding. In humans fMRI analysis of activity across cortex, but especially localized to PFC regions can provide a direct measure of the extent to which macroscale activity patterns match these predictions.

While our tasks was specifically built to probe human behavior, another profitable route could be to re-develop the tasks for an animal model. A primate or rodent model would allow us to directly measure neural activity at the cellular or even sub-cellular level in animals performing the task. In the CDL task, we can then identify how the coordinated action of units gives rise to the emergent circuit, regional and behavioral phenomena observed. In the Cross-Talk task we would be able to explore how features such as stimulus tuning change as the task proceeds at the individual neuron level. Intriguingly our NTK modeling makes the hypothesis that few if any changes to tuning are necessary to generate the patterns of human behavior.

Bibliography

[1] Alan Anticevic and John D. Murray. *Computational psychiatry: mathematical modeling of mental illness*. Academic Press, 2017. 125

[2] David Badre. Cognitive control, hierarchy, and the rostro-caudal organization of the frontal lobes. *Trends Cogn Sci*, 12(5):193–200, May 2008. 126

[3] Ben Baker, Benjamin Lansdell, and Konrad Kording. A philosophical understanding of representation for neuroscience. *arXiv*, 2021. 123

[4] Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Curr Opin Neurobiol*, 46:1–6, 10 2017. 124

[5] Rachael K Blackman, David A Crowe, Adele L DeNicola, Sofia Sakellaridi, Angus W MacDonald, 3rd, and Matthew V Chafee. Monkey prefrontal neurons reflect logical operations for cognitive control in a variant of the ax continuous performance task (ax-cpt). *J Neurosci*, 36(14):4067–79, Apr 2016. 126

[6] Rachael K Blackman, Angus W MacDonald, and Matthew V Chafee. Effects of ketamine on context-processing performance in monkeys: a new animal model of cognitive deficits in schizophrenia. *Neuropsychopharmacology*, 38(11):2090–100, Oct 2013. 126

[7] Randy L Buckner, Jessica R Andrews-Hanna, and Daniel L Schacter. The brain's

default network: anatomy, function, and relevance to disease. *Ann N Y Acad Sci*, 1124:1–38, Mar 2008. 126

[8] James Cavanaugh, Ilya E. Monosov, Kerry McAlonan, Rebecca Berman, Mitchell K. Smith, Vania Cao, Kuan H. Wang, Edward S. Boyden, and Robert H. Wurtz. Optogenetic inactivation modifies monkey visuomotor behavior. *Neuron*, 76(5):901–907, Dec 2012. 124

[9] Thomas B Christophel, P Christiaan Klink, Bernhard Spitzer, Pieter R Roelfsema, and John-Dylan Haynes. The distributed nature of working memory. *Trends Cogn Sci*, 21(2):111–124, Feb 2017. 127

[10] Charlotte A Chun, Laurita Ciceron, and Thomas R Kwapil. A meta-analysis of context integration deficits across the schizotypy spectrum using ax-cpt and dpx tasks. *J Abnorm Psychol*, 127(8):789–806, Nov 2018. 126

[11] Anne G E Collins and Michael J Frank. Cognitive control over learning: creating, clustering, and generalizing task-set structure. *Psychol Rev*, 120(1):190–229, Jan 2013. 125

[12] Christos Constantinidis and Emmanuel Procyk. The primate working memory networks. *Cogn Affect Behav Neurosci*, 4(4):444–465, Dec 2004. 126

[13] Alexis Dubreuil, Adrian Valente, Manuel Beiran, Francesca Mastrogiuseppe, and Srdjan Ostojic. Complementary roles of dimensionality and population structure in neural computations. *bioRxiv*, 2021. 124

[14] Christopher R Fetsch, Naomi N Odean, Danique Jeurissen, Yasmine El-Shamayleh, Gregory D Horwitz, and Michael N Shadlen. Focal optogenetic suppression in macaque area MT biases direction discrimination and decision confidence, but only transiently. *eLife*, 7, Jul 2018. 124

[15] Qinglong L Gu, Norman H Lam, Michael M Halassa, and John D Murray. Circuit mechanisms of top-down attentional control in a thalamic reticular model. *bioRxiv*, 2020. 127

[16] Michael M Halassa and László Acsády. Thalamic inhibition: Diverse sources, diverse scales. *Trends Neurosci*, 39(10):680–693, 10 2016. 127

[17] D H Hubel and T N Wiesel. Uniformity of monkey striate cortex: a parallel relationship between field size, scatter, and magnification factor. *J Comp Neurol*, 158(3):295–305, Dec 1974. 123

[18] Jessica A H Jones, Scott R Sponheim, and Angus W MacDonald, 3rd. The dot pattern expectancy task: reliability and replication of deficits in schizophrenia. *Psychol Assess*, 22(1):131–41, Mar 2010. 126

[19] N Kanwisher, J McDermott, and M M Chun. The fusiform face area: a module in human extrastriate cortex specialized for face perception. *J Neurosci*, 17(11):4302–11, Jun 1997. 123

[20] Colin Kehrer, Nino Maziashvili, Tamar Dugladze, and Tengis Gloveli. Altered excitatory-inhibitory balance in the NMDA-hypofunction model of schizophrenia. *Front Mol Neurosci*, 1:6, 2008. 126

[21] John H Krystal, John D Murray, Adam M Chekroud, Philip R Corlett, Genevieve Yang, Xiao-Jing Wang, and Alan Anticevic. Computational psychiatry and the challenge of schizophrenia. *Schizophr Bull*, 43(3):473–475, 05 2017. 125

[22] Yinqing Li, Violeta G Lopez-Huerta, Xian Adiconis, Kirsten Levandowski, Soonwook Choi, Sean K Simmons, Mario A Arias-Garcia, Baolin Guo, Annie Y Yao, Timothy R Blosser, Ralf D Wimmer, Tomomi Aida, Alexander Atamian, Tina Naik,

Xuyun Sun, Dasheng Bi, Diya Malhotra, Cynthia C Hession, Reut Shema, Marcos Gomes, Taibo Li, Eunjin Hwang, Alexandra Krol, Monika Kowalczyk, João Peça, Gang Pan, Michael M Halassa, Joshua Z Levin, Zhanyan Fu, and Guoping Feng. Distinct subnetworks of the thalamic reticular nucleus. *Nature*, 583(7818):819–824, Jul 2020. 127

[23] Angela M. Licata, Matthew T. Kaufman, David Raposo, Michael B. Ryan, John P. Sheppard, and Anne K. Churchland. Posterior parietal cortex guides visual decisions in rats. *bioRxiv*, 2016. 124

[24] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, Nov 2013. 124

[25] Nikola T Markov, Julien Vezoli, Pascal Chameau, Arnaud Falchier, René Quilodran, Cyril Huissoud, Camille Lamy, Pierre Misery, Pascale Giroud, Shimon Ullman, Pascal Barone, Colette Dehay, Kenneth Knoblauch, and Henry Kennedy. Anatomy of hierarchy: feedforward and feedback pathways in macaque visual cortex. *J Comp Neurol*, 522(1):225–59, Jan 2014. 126

[26] Nicolas Y Masse, Guangyu R Yang, H Francis Song, Xiao-Jing Wang, and David J Freedman. Circuit mechanisms for the maintenance and manipulation of information in working memory. *Nat Neurosci*, 22(7):1159–1167, 07 2019. 124

[27] Edvard I Moser, Emilio Kropff, and May-Britt Moser. Place cells, grid cells, and the brain's spatial representation system. *Annu Rev Neurosci*, 31:69–89, 2008. 123

[28] John D Murray, Alberto Bernacchia, David J Freedman, Ranulfo Romo, Jonathan D Wallis, Xinying Cai, Camillo Padoa-Schioppa, Tatiana Pasternak, Hyojung Seo,

Daeyeol Lee, and Xiao-Jing Wang. A hierarchy of intrinsic timescales across primate cortex. *Nat Neurosci*, 17(12):1661–3, Dec 2014. 127

[29] Evan D Remington, Devika Narain, Eghbal A Hosseini, and Mehrdad Jazayeri. Flexible sensorimotor computations through rapid reconfiguration of cortical dynamics. *Neuron*, 98(5):1005–1019.e5, 06 2018. 124

[30] Blake A Richards, Timothy P Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, Colleen J Gillon, Danijar Hafner, Adam Kepecs, Nikolaus Kriegeskorte, Peter Latham, Grace W Lindsay, Kenneth D Miller, Richard Naud, Christopher C Pack, Panayiota Poirazi, Pieter Roelfsema, João Sacramento, Andrew Saxe, Benjamin Scellier, Anna C Schapiro, Walter Senn, Greg Wayne, Daniel Yamins, Friedemann Zenke, Joel Zylberberg, Denis Therien, and Konrad P Kording. A deep learning framework for neuroscience. *Nat Neurosci*, 22(11):1761–1770, 11 2019. 124

[31] Mattia Rigotti, Omri Barak, Melissa R Warden, Xiao-Jing Wang, Nathaniel D Daw, Earl K Miller, and Stefano Fusi. The importance of mixed selectivity in complex cognitive tasks. *Nature*, 497(7451):585–90, May 2013. 123

[32] Rajeev V Rikhye, Aditya Gilra, and Michael M Halassa. Thalamic regulation of switching between cortical representations enables cognitive flexibility. *Nat Neurosci*, 21(12):1753–1763, 12 2018. 127

[33] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv*, 2017. 123

[34] D Servan-Schreiber, J D Cohen, and S Steingard. Schizophrenic deficits in the processing of context. a test of a theoretical model. *Arch Gen Psychiatry*, 53(12):1105–12, Dec 1996. 126

[35] Hannah R Snyder. Major depressive disorder is associated with broad impairments on neuropsychological measures of executive function: a meta-analysis and review. *Psychol Bull*, 139(1):81–132, Jan 2013. 125

[36] Vikaas S Sohal and John L R Rubenstein. Excitation-inhibition balance as a framework for investigating mechanisms in neuropsychiatric disorders. *Mol Psychiatry*, 24(9):1248–1257, 09 2019. 126

[37] David Sussillo and Omri Barak. Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Comput*, 25(3):626–49, Mar 2013. 124

[38] Daniel LK Yamins and James J DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356, 2016. 123

[39] Guangyu Robert Yang, Michael W Cole, and Kanaka Rajan. How to study the neural mechanisms of multiple tasks. *Curr Opin Behav Sci*, 29:134–143, Oct 2019. 122, 124