

Attractor Dynamics in Large Scale Recurrent Neural Networks

by

Lakshmi Narasimhan Govindarajan

Bachelor of Computing (Hons.), National University of Singapore, 2014

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Cognitive, Linguistic & Psychological Sciences at Brown University

Providence, Rhode Island

February 2023

© Copyright 2023 by Lakshmi Narasimhan Govindarajan

This dissertation by Lakshmi Narasimhan Govindarajan is accepted in its present form by
the Department of Cognitive, Linguistic & Psychological Sciences as satisfying the
dissertation requirement for the degree of Doctor of Philosophy.

Date _____

Thomas Serre, Director

Recommended to the Graduate Council

Date _____

David Badre, Reader

Date _____

Michael J. Frank, Reader

Approved by the Graduate Council

Date _____

Thomas A. Lewis
Dean of the Graduate School

Education	BROWN UNIVERSITY	Providence, RI
2017-2022	<i>Ph.D. in Cognitive Science (Advisor: Thomas Serre), GPA: 4.0</i> • Dissertation: Attractor Dynamics in Large-Scale Recurrent Neural Networks • DARPA Riser 2022 (Early Career Award) • Presidential Award for Excellence in Teaching 2020-21 • Robert J. & Nancy D. Carney Graduate Fellowship Recipient 2021 (Award total: \$60K) • Brains, Minds, & Machines Summer Program 2021 (Woods Hole, MA)	
2010-2014	NATIONAL UNIVERSITY OF SINGAPORE (NUS) <i>(Major) Bachelor of Computing (Hons.) in Computer Science; (Minor) Biophysics</i> • Dissertation: Random Field Methods for Biological Image Segmentation • Turing Fellowship Recipient (Top 3%) Distinguished Undergraduate Research • Certificate of Merit Excellency in the focus area of Algorithms & Theory	Singapore
Experience	AGENCY FOR SCIENCE, TECHNOLOGY, AND RESEARCH (Research Associate)	Singapore
2014-2017	<i>Statistical Machine Learning for Biological Applications PI: Li Cheng</i> • Designed a unified approach for object pose estimation, tracking, and action recognition on manifolds using Lie Group theory • Authored graph-theoretical solutions for analyzing biological tissue samples	
2013-2013	HARVARD MEDICAL SCHOOL (Research Intern) <i>Department of Genetics PI: George Church</i> • Used Co Selection based Multiplexed Automated Genome Engineering (CoS-MAGE) for whole-genome recoding in E. Coli	Boston, MA
2012-2012	BROAD INSTITUTE - HARVARD SEAS (Research Intern) <i>Harvard Laboratory-at-Large PI: Erez Lieberman Aiden</i> • Developed novel computational tools for understanding the structure-function relationship of chromatin and its looping principles	Boston, MA
Publications (Selected)	1. Fast inference of spinal neuromodulation for motor control using amortized neural networks; <i>Journal of Neuroengineering</i> (2022); Govindarajan LN, Calvert JS, Parker SR, Jung M, Darie R, Miranda P, Shaaya E, Borton DA, Serre T. 2. The emergence of visual simulation in task-optimized recurrent neural networks; <i>SVRHM Workshop@NeurIPS</i> (2022); Ashok AK, Govindarajan LN, Linsley D, Sheinberg D, Serre T. 3. A practitioner's guide to improve the logistics of spatiotemporal deep neural networks for animal behavior analysis; <i>Visual observation and analysis of Vertebrate And Insect Behavior workshop, ICPR</i> (2022); Govindarajan LN, Kakodkar R, Serre T. 4. Using Computational Analysis of Behavior To Discover Memory-Guided Attention Mechanisms (<i>PsyArXiv</i> 2021) Amso D, Govindarajan LN, Gupta P, Baumgartner H, Lynn A, Gunther K, Placido D, Sharma T, Veerabadran V, Thakkar K, Kim S, Serre T 5. Tracking Without Re-recognition in Humans and Machines; <i>NeurIPS</i> (2021); Linsley D, Malik M, Kim J, Govindarajan LN, Mingolla M, Serre T. 6. Stable and expressive recurrent vision models; <i>NeurIPS '20 (Spotlight)</i>; Linsley D*, Ashok AK*, Govindarajan LN*, Liu R, Serre T (* equal contribution)	
2017 - 2022		

7. Discriminating between sleep and exercise-induced fatigue using computer vision and behavioral genetics; *Journal of Neurogenetics* (2020) Schuch KN*, **Govindarajan LN***, Guo Y, Baskoylu SN, Kim S, Kimia B, Serre T* & Hart AC* (* equal contribution)
8. Likelihood Approximation Networks (LANs) for Fast Inference of Simulation Models in Cognitive Neuroscience, *ELife* (2021) Fengler A, **Govindarajan LN**, Chen T, Frank M
9. Encoder-Decoder Neural Architectures for Fast Amortized Inference of Cognitive Process Models, *CogSci*, Toronto, (2020) Fengler A, **Govindarajan LN**, Frank M
10. Fully automated leg tracking of Drosophila neurodegeneration models reveals distinct conserved movement signatures, *PLoS biology* 17(6) (2019) Wu S, Tan KJ*, **Govindarajan LN***, Stewart JC, Gu L, Ho JWH, Katarya M, Wong BH, Tan EK, Li D, Claridge-Chang A (* equal contribution)
11. Neural Computing on a Raspberry Pi: Applications to Zebrafish Behavior Monitoring, *Visual observation and analysis of Vertebrate And Insect Behavior workshop, ICPR*, Beijing, China (2018) **Govindarajan LN**, Sharma T, Colwill R, Serre T
12. Robust pose tracking with a joint model of appearance and shape, *arXiv preprint arXiv:1806.11011* (2018) Guo Y, **Govindarajan LN**, Kimia B, Serre T
13. Lie-X: Depth image-based articulated object pose estimation, tracking, and action recognition on lie groups, *International Journal of Computer Vision* (2017) Xu C, **Govindarajan LN**, Zhang Y, Cheng L
14. Hand action detection from ego-centric depth sequences with error-correcting Hough transform, *Pattern Recognition* (72) (2017) Xu C, **Govindarajan LN**, Cheng L

**Posters and
Abstracts
(Selected)
2017 - 2022**

- Taxonomizing the Computational Demands of Videos Games for Deep Reinforcement Learning Agents, *Conference on Cognitive Computational Neuroscience* (2022); **Govindarajan LN***, Liu R*, Ashok AK, Reuter M, Frank MJ, Linsley D*, Serre T.
- Toward modeling visual routines of object segmentation with biologically inspired recurrent vision models, *Vision Sciences Society* (Oral, 2022); Goetschalckx L*, Zolfagher M*, Ashok AK, **Govindarajan LN**, Linsley D, Serre T.
- Leveraging increased electrode density to facilitate optimal epidural spinal stimulation location discovery, *Society for Neuroscience* (2022); Parker SR*, **Govindarajan LN***, Calvert JS, Darie R, Shaaya E, Serre T, Borton DA.
- Evidence accumulation by single units in the human Medial Temporal Lobe (MTL) during memory-based decisions, *Society for Neuroscience* (2022); Yebra M, Schjetan AGP, **Govindarajan LN**, Mosher C, Salimpour Y, Valiante TA, Kalia S, Anderson W, Mamelak A, Rutishauser U.
- Mediolateral Spinal Local Field Potentials Enable Real-Time Decoding of Gait Cycle in an Ovine Model, *Society for Neuroscience* (2021); Parker SR, Calvert JS, Darie R, Shaaya E, **Govindarajan LN**, Jung M, Serre T, McLaughlin B, Borton DA.
- The recurrent neural circuits of contextual effects in visual cortex, *Cosyne* (2021) Linsley D, **Govindarajan LN**, Serre T
- Using Computational Analysis of Behavior to Discover Memory-Guided Attention Mechanisms During Naturalistic Visual Search in Childhood, *Society for Research in Child Development* (2021) Amso D, **Govindarajan LN**, Serre T.
- Uncovering the circuit mechanisms that shape contextual phenomena with task-optimized recurrent neural networks, *VSS* (2021), Linsley D, **Govindarajan LN**, Serre T
- Effects of response-contingent events on behavior in zebrafish larvae, *The 12th Zebrafish Disease Models Conference* (2019), **Govindarajan LN**, Serre T, Colwill R

Patents

- Recurrent neural circuits; US Patent App. 16/868,392; Linsley D, Kim J, Serre T, Ashok AK, **Govindarajan LN**, Liu RG
- Zebrafish pose estimation and tracking using a 3d curve representation; SG Patent App. 10201600799U; **Govindarajan LN**, Stewart J, Cheng L, Chang AC, Jesunathan S, Mathuru AS
- An Automated Machine-Learning System to Track Arthropod Leg Movements and Its Use to Measure Fly Tremors; SG Patent App. 10201707833S; Cheng L, Shuang W, Sherry AW, **Govindarajan LN**

Teaching

- **CLPS0950:** Introduction to Programming | Spring '21, Spring '20, Fall '18 | Brown University
- **CLPS1150:** Memory and the Brain | Spring '18 | Brown University
- **CS1020:** Data Structures & Algorithms | Spring '11 | National University of Singapore

Talks

- Stable and Expressive Recurrent Neural Networks, *Brown Unconference* (Providence, June 2020)
- Deep learning for \$150? *Data Science, Computing, and Visualization (DSCoV) workshop series* (Providence, October 2019)
- AI and Creativity, *A Brown-RISD Collaboration* (Providence, April 2019)
- Neural Computing for the Behavioral Sciences, *Human Brain Colloquium* (Prov.College, Feb 2019)
- Deep Neural Nets for Brain and Behavior Analysis, *Neuroengineering Summer School* (Salve Regina University, Sep 2018)

Awards and Honors

- Early Career Award | U.S. Defense Advanced Research Projects Agency (2022)
- Presidential Award for Excellence in Teaching | Brown University (2021)
- Certificate of Merit (Excellency in Algorithms & Theory) | National University of Singapore (2014)
- Turing Fellow (Distinguished Undergraduate Research) | National University of Singapore (2011 - 14)
- NUS Travel Scholarship | For research conducted at the Koch Institute at MIT (2011)
- International Schools Software Award | South East Asia Regional Computer Confederation (2009)
- National Young Talent Award in Computer Programming | Computer Society of India (2009)
- Computer Literacy Foundation Award | International Informatics Olympiad National Rank 2 (2007)

Service

- **Reviewer** | Conference on Neural Information Processing Systems | 2022
- **Reviewer** | International Conference on Machine Learning | 2021
- **Reviewer Awardee** | International Conference on Learning Representations | 2020
- **Student Organizer** | Perception & Action Seminar Series (Brown University) | 2018 - 2019
- **Journal Club Lead** | Beyond Deep Learning Workshop (Brown University) | 2018
- **Reviewer** | Association for the Advancement of Artificial Intelligence (AAAI) | 2017
- **President and Violinist** | NUS Indian Instrumental Ensemble | 2013

Acknowledgements

Every journey is its own story. And every story has facets that are unobservable to anyone but the narrator. As the narrator of this journey that culminates in the writing of this dissertation, I have the chance to show the world my gratitude to everyone who played a part in turning this dream into reality.

Foremost of them is my advisor and mentor, Thomas Serre. You believed in me during one of the toughest moments in my life. Without your trust and empathy, there would have been no story to tell and no journey to experience, and for that, I am forever grateful. It is also my honor to have been mentored by David Badre and Michael Frank along the way. David, you were my first teacher here at Brown and one from whom I learned a great deal about the art of teaching. I am also indebted to you for all the non-obvious support you provided me, especially when it came to the clutch. Michael, I've come to you with several silly ideas over the years, but you've always received them with the same enthusiasm. You taught me the philosophy of staying balanced yet excited about science, and I'll make sure to pay that forward. I've had the privilege of inspiring collaborators David Borton, Anne Hart, Dima Amso, Justin Fallon, and Ruth Colwill, who shaped me as a generalist and have directly and indirectly impacted this dissertation.

A large part of this journey was spent in a basement and some in the sunlight. I have no doubt that the personal and professional relationships forged during this period provided me the fuel to carry this effort over the line. For having these people in my life, I can only feel lucky. In no small amount, I am deeply indebted to Drew, who challenged me when I felt special and propelled me when I was down. To Alekh, who was in equal parts an inspiration and an emotional pillar. To Alex and Tiantian, who constituted a home away from home. To Ivan, who, no matter what, helped me smile. To Lore, Jaeson, Jonathan, Sam, and Kelsey for being comrade-in-arms. To Jovan, for being the younger brother I needed. And to Kaushik, whose love bore no bounds.

This journey did not begin in Providence. In fact, it started in a very different part of the world, with a different intended destination. I owe it to my sister, Barghavi, and my brother-in-law, Venkat, for introducing me to the world of science. Barghavi, you made this happen. In the years of soldiering that followed, I have been downright fortunate to have *that* blanket of unconditional love and support. My brother, Mukundhan, who taught me that being kind far outweighs everything else in life.

Sowmya and Prashanth, whose influence on me runs deep and knowing they are there made life a lot easier. And, my best friend and champion, Dale, who has tirelessly pushed me to become better and has been singularly responsible for creating an illusion that everything will be okay. I will forever be thankful.

I am here writing this dissertation only because of the sacrifices my family made for me. I *know* the things that my aunts, uncles, my sister, my parents, and my grandparents gave up just so I never felt the wrath of life and could pursue my dreams. Amma and Appa, I hope I've made you proud. Thatha and Paati, I know you are smiling now.

Here, one journey ends, and a new one begins.

Contents

List of Tables	xii
List of Figures	xiii
1 General Introduction	2
1.1 The organization of visual cortex	2
1.2 Computational accounts of feedforward processing	5
1.3 Contextual modulation of visual function	9
1.4 Computation through dynamics	13
1.5 Goal of this thesis	16
1.6 References	18
2 Relevant Mathematical Background	27
2.1 Some basic definitions	27
2.2 Linear stability of ODEs: The 1-Dimensional scenario	29
2.3 Linear stability of vector-valued ODEs	29
2.4 Dynamics of 1-Dimensional maps	31
2.5 Asymptotic behavior of linear dynamical systems	32
2.6 Linear stability of nonlinear dynamical systems	33
2.7 Contraction maps	34
2.8 References	36
3 The Balance between Flexibility and Stability	37
3.1 Synopsis	37
3.2 Introduction	38
3.3 Background	39
3.3.1 Do recurrent CNNs pass the constraint-qualifications test of RBP?	40
3.4 Stabilizing expressive recurrent CNNs	43
3.4.1 Stable and expressive recurrent CNNs with constant memory complexity	44
3.4.2 Microsoft COCO Panoptic Segmentation	47
3.4.3 Related work	48

3.5	Limitations and Future Directions	48
3.6	Conclusion	49
3.7	Extended background	50
3.7.1	Backpropagation through time (BPTT)	50
3.7.2	Lipschitz Coefficient Penalty	50
3.8	Recurrent Back-prop	52
3.9	Recurrent vision models	52
3.9.1	<i>Pathfinder</i>	52
3.9.2	State space analysis	54
3.9.3	Panoptic Segmentation	54
3.10	Extended discussion	55
3.10.1	Related work	55
3.11	References	65
4	The Utility of Visual Representation Dynamics in Deep RNNs	70
4.1	Synopsis	70
4.2	Use case I: Modeling human behavior in a serial grouping task	71
4.2.1	Methods	71
4.2.2	Results	72
4.3	Use case II: Studying the mechanisms underlying visual simulation	74
4.3.1	Methods	74
4.3.2	Results	75
4.3.3	Conclusion	75
4.4	Use case III: Efficient decision-making in reinforcement learning agents	77
4.4.1	Introduction	77
4.4.2	Related work	79
4.4.3	The Learning Challenge Diagnosticator	80
4.4.4	Validating LCD taxonomies by building better agents	83
4.4.5	Discussion	88
4.4.6	Reproducibility statement	88
4.4.7	Appendix	88
4.5	References	103
5	Attractor Dynamics for Information Routing	109
5.1	Introduction	109
5.2	Activity-silent mechanisms and its affordances	111
5.3	Synaptic modulation in feedback networks	116
5.3.1	Methods	116
5.3.2	Results	119
5.4	References	123

6	Discussion and Conclusion	127
6.1	References	131

List of Tables

4.1	Outlining the “Semantic” organization of each environment in the <i>Procgen</i> benchmark. Part I	99
4.2	Outlining the “Semantic” organization of each environment in the <i>Procgen</i> benchmark. Part II	100
4.3	Outlining the “Figure/Ground” organization of each environment in the <i>Procgen</i> benchmark. Part I	101
4.4	Outlining the “Figure/Ground” organization of each environment in the <i>Procgen</i> benchmark. Part II	102

List of Figures

1.1	A hierarchical model of cortical organization. (a) Anatomical characterization of the differences between feedforward, feedback, and lateral connectivity in the cortex. (b) Visual areas in this model are arranged by a simple rule: lower areas project to higher areas through feedforward connections while the reverse happens through feedback connections. (Modified from [3]) (c) Visual areas sorted per their response latencies. Ticks on each line denote the 10 th , 50 th , and 90 th percentiles. Each reference in parantheses corresponds to those listed in [4]. Overall figure adapted from [4, 5]. .	3
1.2	Feedforward models of the ventral pathway. (a) Hubel and Weisel’s original hierarchical formulation that accounts for simple and complex cell responses to oriented bars in the cat striate cortex. Adopted and modified by author from [14] (b) The HMAX model extends Hubel and Weisel’s proposal. The non-linear max pooling operator in this model helps reconcile physiological and behavioral observations, especially in the case of viewpoint and scale invariance, and has a plausible cortical circuitry implementation. Adopted from [45] (c) More recently, deep convolutional networks (DCNs) have cemented their status as the dominant models of feedforward processing and excel at several large-scale visual challenges. Depicted here is a popular neural architecture, VGG16. Adopted from [46].	6
1.3	Contextual modulation of behavioral and neural responses. (a) The perceived color of the central orange ring is modulated by the color of the rings surrounding it. A purple surround strikingly shifts the perceived color of the central ring towards the pink end of the spectrum. Adopted and modified from [87]. (b) Multi-unit recordings from V1 revealed that surround suppression is dynamics and depended on the degree to which the center and surround stimuli match. Adopted from [88]. (c) In a more nuanced experiment, [88] find that surround suppression is, in fact, feature-specific and the suppression acts component-wise. Adopted from [88].	10

1.4	Recurrent neural network models of primary visual cortex (a) The proposed model of Mely et al. subscribed to a hyper-columnar architecture that distinguished near and far eCRF surrounds that had complementary effects on the CRF responses. The model reconciled a host of eCRF phenomenon and provided mechanistic insights into psychophysical measures of human behavior. Adopted from [87] (b) The horizontal gated recurrent unit (hGRU) from Linsley et al. [104] is a machine learning module that implements mechanisms described in [87]. The parameters of this unit, however, are learnable through gradient descent. Recurrent activities are denoted by H , kernels by W , input by X , and gates by U . $+$ and $\times$ symbolize linear and quadratic interactions respectively. Adopted from [104].	12
1.5	A toy illustration of neural population “states” and dynamics. (a) Consider a population consisting of three neurons. In any given interval of time (vertical, gray dashed lines), individual neurons can have varying levels of activity. By summarizing the overall activity of each neuron in a time window (either by binning and counting spikes or by taking the average firing rate), one can effectively represent the “state” of this population by a vector in three dimensions, with one coordinate per neuron. (b) Tracking the evolution of this three dimensional vector through time (i.e., through successive time windows) results in a path that is called the neural trajectory. The gray shaded region denotes the subspace of $\mathbb{R}^3$ in which the activity patterns from this example reside.	15
2.1	Visualizing the dynamics of 1-D maps using cobweb plots. Cobweb plots are a simple visualization strategy to trace the evolution of the state as we iterate through a map. Here these trajectories are shown as dashed lines. The gray dashed line is the line of unity. (a) The linear system $x_{n+1} = -0.5x_n$ has a fixed point at $x = 0$ that is stable. (b) The linear system $x_{n+1} = -1.2x_n$ also has a fixed point at $x = 0$ but is unstable. (c) The logistic map is a 1-D non-linear system. Here we show the dynamics for the map $x_{n+1} = 2.5x_n(1 - x_n)$. The fixed point at $x = 0$ is unstable, while the one at $x = 1$ is stable.	32

2.2	Linear stability of equilibria in a 2-D dynamical system	The local dynamics of a system around a fixed point is fully specified by the eigenvalues of its Jacobian matrix. (a) In the case of real eigenvalues, we have four common scenarios. When both the eigenvalues are of magnitude greater than 1, we have unstable local dynamics, and when both have a magnitude smaller than 1, we have stable dynamics. In the other two scenarios, we have what's called a saddle point. This generally means that the dynamics attract along one axis and repel along the other. In the special case of either one or both eigenvalues equal to 1, we need to explicitly check for Lyapunov stability. (b) In the case of complex conjugate eigenvalues, we have rotational dynamics around the fixed point. Whether it's an attracting or repelling spiral depends on the magnitude of the complex vector. If the magnitude of the complex eigenvalue is equal to 1, then we have stable, fixed amplitude rotations.	34
3.1	Recurrent CNNs trained with backpropagation through time (BPTT) have unstable dynamics and forget task information.	This pathology is corrected by our <i>Lipschitz Coefficient Penalty</i> (LCP). (a) Incremental segmentations of horizontal gated unit (hGRU) models trained for N recurrent steps on Pathfinder-14. Heatmaps in the left column show the models tracing the target contour until $t=N=6$ steps. The right column depicts model predictions after $t=T=40$ steps. Segmentations from the BPTT-trained hGRU degenerate but models trained with LCP did not. Learning algorithms of LCP-trained models are “contractor-BPTT” (C-BPTT) and “contractor-RBP” (C-RBP). (b) Visualization of horizontal gated unit (hGRU) state spaces by projecting hidden states onto each model's top-two eigenvectors. Grey dots are the 2D-histogram of projected hidden states, red contours are hidden state densities up to the task-optimized N steps, and blue contours are hidden state densities following that step. Dynamics for an exemplar image are plotted in yellow. BPTT-trained model dynamics diverge when $t > N$, but models trained with LCP did not. (c) Two-sample KS-tests indicate that the distance in state space between $t = N$ and $t = T$ hidden states is significantly greater for an hGRU trained with BPTT than an hGRU trained with C-BPTT or C-RBP (n.s. = not significant).	42

3.2	Enforcing contraction in recurrent CNNs improves their performance and parameter efficiency and enables our constant-memory C-RBP learning algorithm. (a) hGRU models were trained and tested on different versions of <i>Pathfinder</i> . Only the version trained with C-RBP maintained high performance across the three datasets. (b) C-RBP models can rely on recurrent processing rather than spatially broad kernels to solve long-range spatial dependencies. BPTT-trained models cannot practically do this due to their linear memory complexity. (c) LCP improves the stability of hGRU dynamics and, as a result, the generalization of learned visual routines for contour integration. Models were trained on Pathfinder-14, and tested on all three Pathfinder datasets. hGRUs trained with C-RBP and C-BPTT generalized far better than a version trained with BPTT or a 6-layer CNN control. Numbers above each curve denote the max-performing step.	44
3.3	C-RBP trained recurrent vision models outperform the feedforward standard on MS-COCO Panoptic Segmentation despite using nearly 800K fewer parameters. (a) Performance of our recurrent FPN-ResNet 50 trained with C-RBP improves when trained with more steps of processing, despite remaining constant in its memory footprint. (b) Recurrent processing refines instance segmentations and controls false detections exhibited by the FPN-ResNet 50 (additional examples in SI). (c) Panoptic segmentation timecourses for an FPN-ResNet 50 trained with C-RBP for 20 steps.	46
3.4	C-RBP-optimized models for Panoptic segmentation learn to implement a routine that resembles a flood-fill algorithm, despite no explicit supervision to do so. Top Flood-filling in a Voronoi-tesselation of object masks. Bottom The routine learned by a C-RBP R-FPN-ResNet-50.	49
3.5	Pytorch 1.4 code for computing the LCP.	51
3.6	Parameterically controlling task difficulty. In our variation of the <i>Pathfinder</i> challenge [7, 8], we ask observers to segment the contour connected to the white dot. Recurrent CNNs can easily solve it by learning to incrementally “trace” the target from end to end. Target contours are made up of 14-, 20-, or 25-dashes.	53

3.7	Convolutional LSTMs trained with (BPTT) exhibit unstable dynamics, like the BPTT-trained hGRUs. Once again, LCP corrects this pathology. (a) Visualization of convLSTM and hGRU state spaces following the state space method described in Section . Here, the BPTT-LSTM was trained for 6 steps, the C-RBP LSTM for 60 steps, and the C-RBP hGRU for 40 steps. Grey dots are the 2D-histogram of projected hidden states, red contours are hidden state densities up to the task-optimized N steps, and blue contours are hidden state densities beyond that point ($t > N$). Exemplar dynamics for a single image are plotted in yellow. While dynamics of the BPTT trained model diverge when $t > N$, models trained with LCP did not. (b) Model dynamics are reflected in their performance on Pathfinder-14 at $t = N$ and $t = T$ steps of processing. (c) Two-sample KS-tests indicate that the distance in state space between $t = N$ and $t = T$ hidden states is significantly greater for the BPTT-trained convLSTM than for either of the models trained with C-RBP (n.s. = not significant).	56
3.8	Additional state space analyses showed that alternatives to BPTT do not resolve the unstable dynamics we observed for recurrent CNNs. Here, BPTT per-step supervision refers to a model which was optimized with a loss evaluated on each of its 6 steps of processing. T-BPTT refers to a model trained with truncated backprop, for which gradients were accumulated over 3 steps of its 6 steps of processing. (a,b,c) These BPTT alternatives train models with unstable dynamics, which forgot task information after the optimized $t = N$ steps of processing. The distances between $t = N$ and $t = T$ hidden states are significantly greater for hGRUs trained with these algorithms than for an hGRU trained with C-RBP (n.s. = not significant).	57
3.9	Evaluating model performance. Exemplars from the (a) <i>Pathfinder</i> challenge, along with (b) model performance on each of these datasets, and (c) predicted contours from hGRUs trained with the different algorithms.	58
3.10	Performance of hGRUs during training on <i>Pathfinder</i> challenge datasets. The RBP-trained model struggles to fit any dataset, unlike the models trained with BPTT, C-BPTT, or C-RBP.	58
3.11	Tracking LCP. The value of our LCP (computed with Eq. 6 in the main text) over the course of training for models that minimize it (C-RBP, C-BPTT) and models that do not (RBP, BPTT). In other words, the magnitude of this correlates with the stability/instability of model dynamics.	59

3.12	Generalization performance for hGRUs and convLSTMs trained with BPTT and alternatives to BPTT. Models were trained on Pathfinder 14 and tested on Pathfinder 14/20/25. For reference, performance of the hGRU trained with C-RBP is plotted in both rows. BPTT per-step loss means that a loss was computed on each of the 6 steps of hGRU training, and weights were optimized with BPTT. In contrast, a loss was only calculated on the final step for BPTT. TBPTT is truncated backprop through time, where gradients were computed over 3 steps of the 6-step dynamics. The LSTM trained with BPTT was trained for 6 steps, whereas the LSTM trained with C-RBP was trained for 60.	60
3.13	Recurrent model performance on MSCOCO Panoptic Segmentation. Performance was computed for each of 30 steps of processing for models trained with BPTT and C-RBP. The C-RBP models achieve better and more stable performance. . . .	61
3.14	Panoptic segmentation performance. Our recurrent Panoptic Segmentation models also outperform the feedforward standard when both are given ResNet-101 backbones and the $3\times$ training schedule (see https://bit.ly/dtcon for details on this training routine). The C-RBP model, trained for 20 steps, outperforms any other tested version of the model.	62
3.15	Panoptic predictions from the feedforward ResNet-50 FPN Mask-RCNN (left) and our recurrent version of the model trained with C-RBP (right).	63
3.16	Panoptic predictions from the feedforward ResNet-50 FPN Mask-RCNN (left) and our recurrent version of the model trained with C-RBP (right).	64
4.1	Activity dynamics of an hGRU trained to solve the COCO-Dots challenge recapitulate human decision-making time courses. We observe that the hGRU’s internal activity dynamics are a function of the cue and fixation dot configuration. By considering the first timepoint at which activations from the cue and fixation dots merge as a proxy for the model reaction time, we can reliably correlate the time courses of model behavior and human decision-making. In agreement with human psychophysics data presented in Jeurissen et al. [1], we find that our models are fast when the two dots are close to each other and slow when they are either far apart or one of them resides in a narrow region in space.	72
4.2	Visual representation dynamics in our feedback model, InT, reveal a simulation strategy. (a) Exemplar inputs to the InT circuit. For illustration purposes, here we hold all components of a <i>Planko</i> board constant while only manipulating the ball position. (b) Visualizing the internal dynamics of the InT model right before it makes a decision reveals an evidence accumulation process. (c) “Simulated” ball trajectories can be decoded from the latent activity space of the feedback model. Visual simulation emerges as an optimal solution in this case, as the feedback circuit was never explicitly guided to perform this computation.	76

4.3	The Learning Challenge Diagnosticator (LCD) separately measures the reinforcement learning or the perceptual challenge of an environment and task. (a) We use LCD to compute perceptual and reinforcement learning challenges in each game of the <i>Procgen</i> Benchmark. (b) Perceptual challenges are assessed by comparing the total reward accrued by agents that learn over visual representations with different amounts of structure: pixels, figure/ground segmentation masks, and semantic segmentation masks (see Appendix. 4.4.7). (c) Reinforcement learning challenges are computed by feeding agents perceptually organized scenes (i.e., semantically segmented; second and third rows), and then comparing the total reward accrued by one agent playing in an environment with stochastic manipulations of reward (randomly masked and modulated; third row) to another agent playing in the normal environment (second row). (d) The taxonomy of <i>Procgen</i> reveals each game’s relative perceptual and reinforcement learning challenge.	78
4.4	A DNN that is self-supervised to predict future frames also learns a concomitant representation of object-ness that is useful for reinforcement learning. (a) Our routine for self-supervising perceptual representations that can solve challenges identified by the LCD utilizes three networks: a ResNet18 [1] for predicting the optic flow between frame t to frame $t + 1$ (<i>Flow encoder</i>), a ResNet18 feature pyramid network (FPN) with a 16-channel recurrent neural network [2] to encode the content of frame t (<i>Object encoder</i>), and an FPN-ResNet18 decoder that receives the two encoder’s outputs to transform frame t to frame $t + 1$ via a differentiable warping module. After optimizing for next-frame prediction, the <i>Object encoder</i> learns object-like representations that can be used for reinforcement learning without additional training. This approach generates reasonable segmentations of the important objects in <i>Procgen</i> games, including Bigfish, Starpilot (b) , and Bossfight (c)	81
4.5	Self-supervised pre-training for learning object-ness addresses perceptual challenges in <i>Procgen</i> identified by the LCD. (a) We measured the impact of our self-supervised perceptual front-end for learning games classified as having perceptual challenges (Hard ϕ) or not (Easy ϕ). The game which is helped the least by the front-end, “Leaper”, requires agents to recognize colors – a feature that is not preserved in the front-end. (b) When looking at aggregate performance across the entirety of <i>Procgen</i> , the benefit of pre-training is only seen on those games that the LCD deems as perceptually challenging (Hard ϕ), indicating that the current state of progress in dRL is being slowed by an inability to find benchmarks (or subsets of benchmarks) which are aligned with the algorithmic solutions being tested. One-tailed t -tests assessing the effect of learning from the perceptual front-end versus pixels are denoted by lines, $* = p < 0.05$	83

4.6	LCD measurements of the perceptual and reinforcement learning challenges in <i>Procgen</i> translate across dRL algorithms. Measurements of visual challenges ϕ (left) and reinforcement learning challenges ψ (right) faced by PPO agents were significantly correlated with the challenges of PPG agents, indicating that results derived from applying the LCD to simpler and lower-overhead algorithms can guide the development of agents pushing the state-of-the-art.	84
4.7	A subset of games in <i>Procgen</i> see a synergistic benefit from co-training perception and action/policies end-to-end. We compared the magnitude of improvement over baseline (PPO/pixels) agents experienced when they were given an ideal reinforcement learning algorithm (PPG) and perceptual representations (semantic segmentation) to the additive improvement of an agent given PPG over pixels and an agent given PPO and semantic segmentation inputs. Only 6/16 games benefit from co-training.	87
4.8	Normalized reward curves while systematically varying perceptual complexity. X-axis denotes the number of “interactions” an agent performs with its environment during training and Y-axis denotes normalized rewards. We trained all these agents for a total of 200M steps.	91
4.9	Normalized reward curves while systematically varying reward stochasticity. X-axis denotes the number of “interactions” an agent performs with its environment during training and Y-axis denotes normalized rewards. We trained all these agents for a total of 200M steps. The perceptual input to these agents were semantic representations. While performing evaluations, the reward mechanisms were set to the original configuration thus effectively only testing the impact of these perturbations on policy learning.	92
4.10	Normalized reward curves, on <i>i.i.d.</i> generalization levels, while systematically varying perceptual complexity. X-axis denotes the number of “interactions” an agent performs with its environment during training and Y-axis denotes normalized rewards. We trained all these agents for a total of 200M steps. Agents were evaluated approximately every 60K steps.	93
4.11	Normalized reward curves, on <i>i.i.d.</i> generalization levels, while systematically varying reward stochasticity. X-axis denotes the number of “interactions” an agent performs with its environment during training and Y-axis denotes normalized rewards. We trained all these agents for a total of 200M steps. The perceptual input to these agents were semantic representations. Agents were evaluated approximately every 60K steps. While performing evaluations, the reward mechanisms were set to the original configuration thus effectively only testing the impact of these perturbations on policy learning.	94

4.12	Training dRL agents with varying levels of perceptual complexity using the proximal policy gradient algorithm. X-axis denotes the number of “interactions” an agent performs with its environment during training and Y-axis denotes rewards on training levels. We trained these agents for a total of 100M steps.	95
4.13	Training dRL agents with varying levels of stochastic feedback using the proximal policy gradient algorithm. X-axis denotes the number of “interactions” an agent performs with its environment during training and Y-axis denotes rewards on training levels. We trained these agents for a total of 100M steps. The perceptual inputs to these agents were semantic representations. While performing evaluations, the reward mechanisms were set to the original configuration thus effectively only testing the impact of these perturbations on policy learning.	96
4.14	Policy learning experiments in dRL agents operating on visual representations learned from natural language supervision. We test the extent to which generalist visual representations support policy learning. We find that while CLIP [76], a transformer-based architecture, is state-of-the-art on zero-shot image classification, its representations do not support policy learning adeptly. We train agents on both the “easy” and “hard” versions of <i>Progen</i> . While our agent was able to learn certain tasks in the “easy” mode, they predominantly struggled in the “hard” mode. X-axis denotes the number of “interactions” an agent performs with its environment during training and Y-axis denotes rewards. We trained these agents for a total of 200M steps.	97
4.15	The effect of “reward shaping” on Leaper – a high ψ task as predicted by our taxonomy. In the traditionally challenging reinforcement learning problem we find that encouraging agents to discover sub-goals significantly improved both the sample efficiency and generalization capacity ($t = 41.03, p < .001$). X-axis denotes the number of “interactions” an agent performs with its environment during training and Y-axis denotes normalized rewards. We trained both these agents on semantic representations for a total of 200M steps. While performing evaluations, we revert to the original reward scheme to remain compatible for comparing to the “naive” model.	98
5.1	The expressivity of an hGRU module. The hGRU circuit implements Dale’s law and separates neurons into distinct excitatory (h_2) and inhibitory (h_1) sub-populations. It’s learned connectivity structure can allow for recurrent excitation and feedback inhibition. Here we show an example of how a low-dimensional activity manifold of the hGRU can resemble a quasi-stable oscillator. The network parameters driving these dynamics were identified using an approximate Bayesian inference algorithm [10].	110

5.2	Illustrating some of the computational benefits of dynamic synapses. (a) Synaptic potentiation can maintain a “trace” of a network’s input even in the absence of overall activity and can subsequently be recovered through a non-specific probe. (b) Synaptic depression can suppress activity in a sub-network postsynaptic transmission and, in the process, generate a traveling wave-like phenomenon. (c) Similarly, strong facilitation following destabilization can generate damped oscillatory patterns.	112
5.3	Temporal filtering as an emergent property of dynamic synapses. (a) (Top) Postsynaptic potential from a $20Hz$ stimulation through a depressing synapse. Subsequent volleys of presynaptic input evoke diminishing postsynaptic responses. (Middle) Postsynaptic potential from a $20Hz$ stimulation but through a facilitating synapse. Here, one can see a supra-linear effect of superposing volleys of presynaptic activity. (Bottom) Postsynaptic potential from a $70Hz$ stimulation through a facilitating synapse. Figure adapted from [29]. (b, c) Simulating synaptic efficacy (facilitation and depression, respectively) as per the model proposed by [30].	113
5.4	A hybrid, image-computable model of visual manipulation. (a) A ResNet-18 encoder-decoder architecture is trained from scratch to extract latent visual representations from the input shapes. These representations are then fed into an RNN with synapses modulated by STSP. The RNN is tasked with determining if there was indeed a match between the sample and test objects presented on either sides of a delay period. Though the whole architecture is trainable end-to-end, in this study we separate the encoder-decoder training from that of the RNN. (b) Visualizing the latent embeddings from the encoder reveals its smooth but highly non-linear structure. Here we depict how variations in object viewpoint are encoded in the principle components of the latent encoding.	117
5.5	Population decoding analysis on RNNs trained to solve the delayed match to sample task. (a) Sample decoding accuracy from neural activity (in green) and synaptic efficacy (in orange) reveals that a model trained to perform a delayed match to sample task, on a restricted set of sample viewpoints, primarily utilizes synaptic substrates to store information about the sample over the delay period. (b) However, the same model when trained over a wide variety of sample viewpoints adopts a more complex strategy that does not obey a clear dichotomy. Both models successfully solve their respective tasks with accuracies $> 90\%$	120

5.6 Visual decoding of latent RNN representations over the delay period.

Information in the synaptic efficacies of a trained RNN, over the course of a trial, can be visually inspected by our pre-trained ResNet-18 decoder. To do so, we first train a linear map to project synaptic variables into the decoder’s embedding space and subsequently use this transformation to reconstruct the “memory” from synaptic traces. We present an illustrative example to highlight how visual decoding strategies can shed light onto the complex computational mechanisms employed by RNNs with dynamic synapses. 122

Abstract of “Attractor Dynamics in Large Scale Recurrent Neural Networks” by Lakshmi Narasimhan Govindarajan, Ph.D., Brown University, February 2023.

Biological brains are dynamic. Recent advances in electrophysiology and neuroimaging have helped uncover various mechanisms through which brains construct and utilize rich variations in neural activity patterns for computations. In particular, the primate visual cortex consists of more than thirty densely interconnected areas whose constituent neurons exhibit a wide range of complex activation patterns in response to visual stimulation. Despite this, models of visual function have historically embraced a highly simplified paradigm within which computations are carried out as a finite cascade of operations in serial. The deep feedforward neural network (DNN) family is the modern-day manifestation of such a model class. In recent years, significant technological advances in computing infrastructure coupled with the availability of large, annotated datasets have propelled the adoption of DNNs as the *de facto* model of vision. DNNs have indeed demonstrated immense proficiency when trained on large-scale, naturalistic visual challenge benchmarks to the extent that they are now on par with, or sometimes beyond, human-level performance. Their impressive successes, however, mask fundamental deficiencies. Our experiments, probing the ability of DNNs to systematically generalize to held-out parameterizations of *Pathfinder*, a visual challenge task suite, reveal one such deficiency: DNNs can appear to learn fundamental and reusable visual computations while crudely approximating an input-output mapping pertaining to the dataset and task at hand. We posit that this pathology is a consequence of having only a fixed computational budget specified by the number of operations in a model’s computational graph. Recurrently connected networks, like those in the primate visual cortex, have the ability to leverage an infinite computational budget through feedback cycles and are promising candidate solutions for the problem mentioned above. However, recurrent network models are hard to train due to well-documented gradient and memory issues. We draw inspiration from neuroscience and derive a novel learning algorithm called Contractive Recurrent Backpropagation (C-RBP) that relies on our model constructing fixed-point attractors. We demonstrate that recurrent vision models trained with C-RBP can not only learn hard parameterizations of *Pathfinder* but can also successfully generalize to held-out parameterizations at a fraction of parameter- and memory costs when compared to DNNs. We subsequently extend these principles to recurrent vision models trained to solve a large-scale visual challenge (MS-COCO Panoptic Segmentation) and show that our approach outperforms the leading feedforward approach with far fewer free parameters. Furthermore, we show that the visual representation dynamics emerging from our C-RBP trained recurrent neural networks can ably support downstream utilization for decision-making, provide a framework to study the interactions between vision and higher cognitive functions, and facilitate the comparison between model and human behavior in various scenarios.

Chapter 1

General Introduction

The genesis of complex visual function can be traced back to the first few hundred milliseconds following visual onset. Light is converted to an electrical signal by Rhodopsins, photoreceptors in the retina, before being transmitted via the optic nerve to the lateral geniculate nucleus (LGN) in the thalamus. The LGN then relays this information to the primary visual cortex (V1) in the occipital lobe. V1 projects into two primary pathways: the ventral pathway into the temporal lobe and the dorsal pathway into the parietal and prefrontal cortex. By the time information reaches the inferotemporal cortex in the ventral pathway, about $150ms$ from the onset, we can already assess the colors, forms, and shapes of entities in our field of view. Around the same time, information in the dorsal pathway has matured to reflect the locations and velocities of said entities and can inform the motor areas in the brain about appropriate visually-guided actions. Information flows back and forth between higher cortical areas and the visual cortex over the next $300ms$ to shape the neural computations responsible for carrying out complex and composite visual functions, like spotting your cat hiding behind your laundry hamper. And this is just the first $500ms$. Decades of research in the vision sciences and more recent insights from large-scale, data-driven machine learning models of vision have established that the cascade of computations in the first $150ms$ following visual onset can be well-approximated by “feedforward” computations. However, computational-level explanations for how neural dynamics and feedback signals shape visual processing beyond this feedforward sweep is still lacking.

1.1 The organization of visual cortex

The primate visual cortex consists of around 32 distinct brain areas intricately interconnected with each other [1]. More than 50% of the cortex is estimated to be comprised of visual areas, and several decades of research has shed light on their anatomical and functional organization. A prominent neurobiological model of the visual cortex was developed by Van Essen and colleagues based on laminar patterns of connectivity and one clear anatomical principle: information flow from lower order areas to higher order areas happened via “feedforward” connections that were distinct

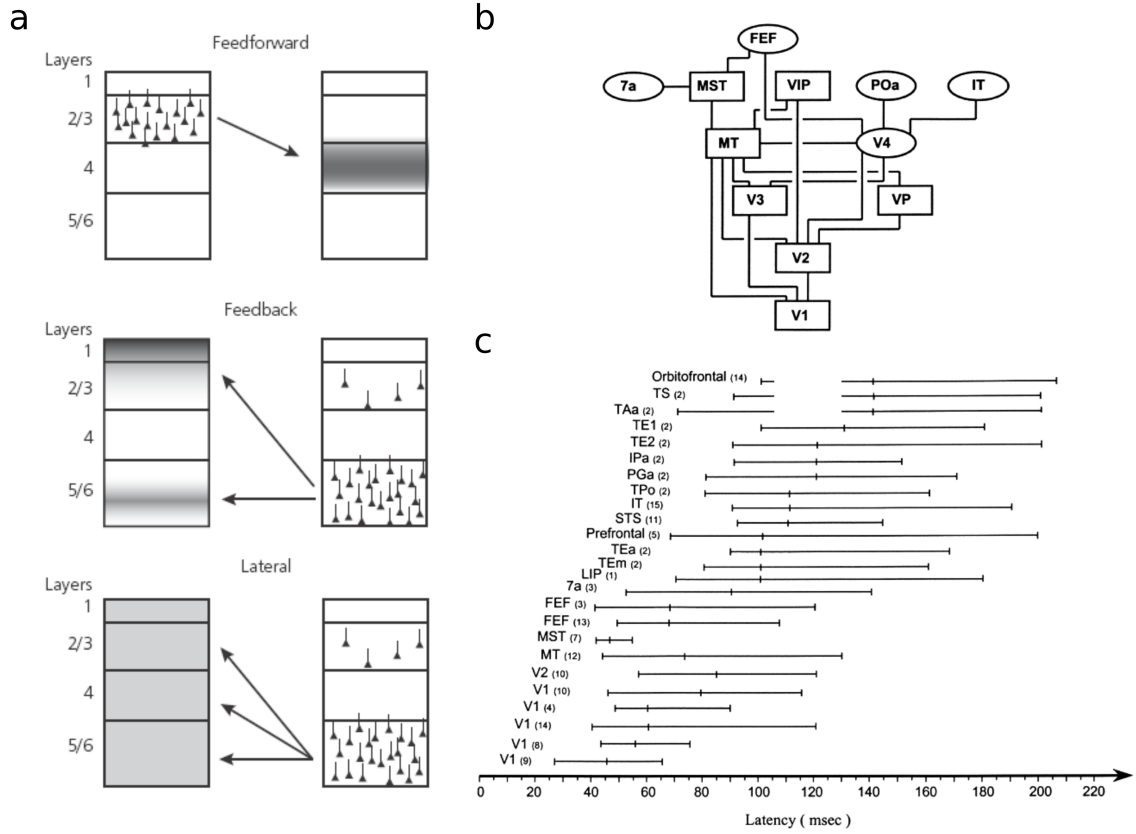


Figure 1.1: **A hierarchical model of cortical organization.** (a) Anatomical characterization of the differences between feedforward, feedback, and lateral connectivity in the cortex. (b) Visual areas in this model are arranged by a simple rule: lower areas project to higher areas through feedforward connections while the reverse happens through feedback connections. (Modified from [3]) (c) Visual areas sorted per their response latencies. Ticks on each line denote the 10th, 50th, and 90th percentiles. Each reference in parantheses corresponds to those listed in [4]. Overall figure adapted from [4, 5].

from the “feedback” connections responsible for carrying information in the reverse direction [2, 3] (Fig. 1.1b). The distinction between feedforward and feedback connections was initially described by Rockland and Pandya, who noted that “forward” projections originated in neurons located in the supragranular layers (L2/3) and terminated around Layer 4, while reciprocal projections originated in the infragranular layers (L5/6) and terminated outside Layer 4 [2] (Fig. 1.1a).

Cortico-cortical connections are thought to be governed by a few general principles [6]: (1) Inter-area projections are primarily made by pyramidal neurons; (2) These pyramidal neurons also spread excitatory input locally, typically within a few millimeters; (3) GABAergic interneurons mediate intra-area projections in addition to the aforementioned local pyramidal input; (4) The density of inter-area connections is proportional to proximity. Formalization of these straightforward rules guided the Van Essen model, later extended to account for newer connectivity data [1].

Primate electrophysiology studies also probed the temporal aspects of interactions between

cortical areas [7–12]. A standard paradigm to study temporal interactions was to measure neural response latency to visual stimulation in different parts of the cortex. Response latencies to visual stimulation (Fig. 1.1c) loosely correlated with the location of these areas in the supposed hierarchy. However, there were notable exceptions to this rule, suggesting that hierarchical ordering alone is insufficient to explain organization [13]. Later it was found that areas with a larger proportion of myelinated axons exhibited these shorter latencies [13]. The discovery of physiological affordances for varying information transfer speeds gave rise to a new paradigm of thought: visual processing could perhaps comprise of distinct temporal phases. Particularly interesting was the suggestion that information was initially rapidly conveyed to higher-order areas to support early computations, which then orchestrated subsequent processing [4].

In the meanwhile, converging evidence from anatomy (connectivity patterns, lesions, neuron types, etc.) and physiology (tuning, receptive field sizes, etc.) helped us better understand the *functional* specialization and organization of the visual areas. Hubel and Weisel’s extraordinary receptive field (RF) mapping efforts provided some of the first physiological evidence for the hierarchical organization model [14, 15]. They found area V1 to have the smallest RFs (lowest in the hierarchy) and that RF sizes increase at successive stages of the hierarchy. Subsequent RF mapping studies [16, 17] established that the RF centers of postsynaptic neurons corresponded to the sum of the RF centers of the afferent feedforward connections. This organizational principle, termed “retinotopicity”, is critical to the spatial representation of visual fields and is now acknowledged as ubiquitous across the cortex [18–20]. Several independent efforts over the years further contributed to our understanding of the functional roles of different visual areas. Neurons in V1 demonstrated selectivity for visual orientation [14], spatial frequency [21], and color [22, 23]. V2 neurons, in addition to V1-like properties, showed selectivity to more complex feature conjunctions such as contours [24, 25] and binocular disparity [26]. V4 selectively responded to object features of intermediate complexity such as simple geometric shapes [27, 28] while the inferotemporal area (IT) responded to more complex object features and faces [29, 30]. The dorsomedial area (DM) responded to visual stimuli associated with self-motion [26] and the middle temporal area (MT) responded to the speed and direction of moving visual stimuli [31, 32].

A breakthrough for functional studies came from identifying two distinct streams separated as early as V1. Visual lesion studies in macaques played an essential role in revealing this dichotomy [33, 34]. Bilateral lesions to area TE in the inferior temporal cortex severely impaired object discrimination ability but not positional knowledge. The opposite effect was found when the posterior parietal cortex was removed bilaterally. Physiologically, the inferotemporal cortex was shown to selectively respond to complex object features [35], while neurons in the middle temporal area were shown to be direction-, speed-, and binocular disparity selective [31, 36, 37]. Ever since, the ventral and dorsal streams have been associated with object and spatial vision, respectively [34].

Despite all this complexity in organization and connectivity, it was becoming increasingly clear that the visual system can process information rapidly [13]. The latency of visually-guided movement, through the dorsal pathway, following stimulus onset was only on the order of $100ms$ [38]. Similarly,

the visual recognition of complex objects and scenes through the ventral pathway took humans only as little as $150ms$ [39]. A theme unifying these observations started to emerge [40, 41]. Visual processing speed relied on a sequence of swift “bottom-up” computations that resulted in coarse visual representations with sufficient information for various functions. Neurophysiological support for this claim came from studies demonstrating that object identity and category information can be “decoded” from local field potentials of neural populations in IT as soon as tens of milliseconds from stimulus onset [42, 43].

A candidate hypothesis for what happens beyond this initial feedforward sweep is that slower feedback processes shape and mediate complex visual routines on top of these base representations. We will review important feedback literature in Section. 1.3 but it is worthwhile noting here that feedback has been generally more challenging to study, mainly since it encompasses a variety of processes, including visuospatial attention and visual working memory. A lot of vision research has focused on identifying the computational basis of feedforward processing and has had a significant impact in both the neuroscience and machine learning spheres. We will now review significant landmarks in this regard.

1.2 Computational accounts of feedforward processing

How do computations carried out by the intricately connected visual areas we discussed above lead to an integrated percept? This question remained a major research area for the better part of the last few decades despite our detailed anatomical knowledge of the visual system [44]. Towards addressing this question, the focus turned to a more tractable subproblem: how can different cell types (such as the simple and complex cells recorded in the cat striate cortex [14]) respond strongly to the same visual stimuli but possess different physiological properties? In this particular case, both simple and complex cells responded to oriented bars but while simple cells had smaller, phase-selective receptive fields, complex cells had larger, phase-invariant receptive fields. One of the first models to suggest a scheme that explains the construction of invariance properties was from Hubel and Weisel (Fig. 1.2a). In this model, simple cells received inputs from neighboring LGN neurons, while complex cells received inputs from neighboring simple cells. The hierarchical arrangement naturally meant that the effective receptive field size of the complex cells was larger than that of simple cells. The phase-invariant response of complex cells was due to the aggregation of inputs from phase-selective simple cells.

The concept of building up selectivity and invariance properties through a hierarchical cascade of computations started to gain popularity in the vision sciences community. Around the same time when Marr philosophized about the gradual increase of representational complexity (from primal to 2.5D to 3D sketches) for shape perception [47], Fukushima’s Neocognitron, a computer vision model, demonstrated the ability to perform translation-invariant object recognition [48]. In similar fashion, Perrett and Oram argued that one way to build invariance to any transformation is to pool over afferents tuned to the different transformed versions of the stimulus [49]. Though the Neocognitron

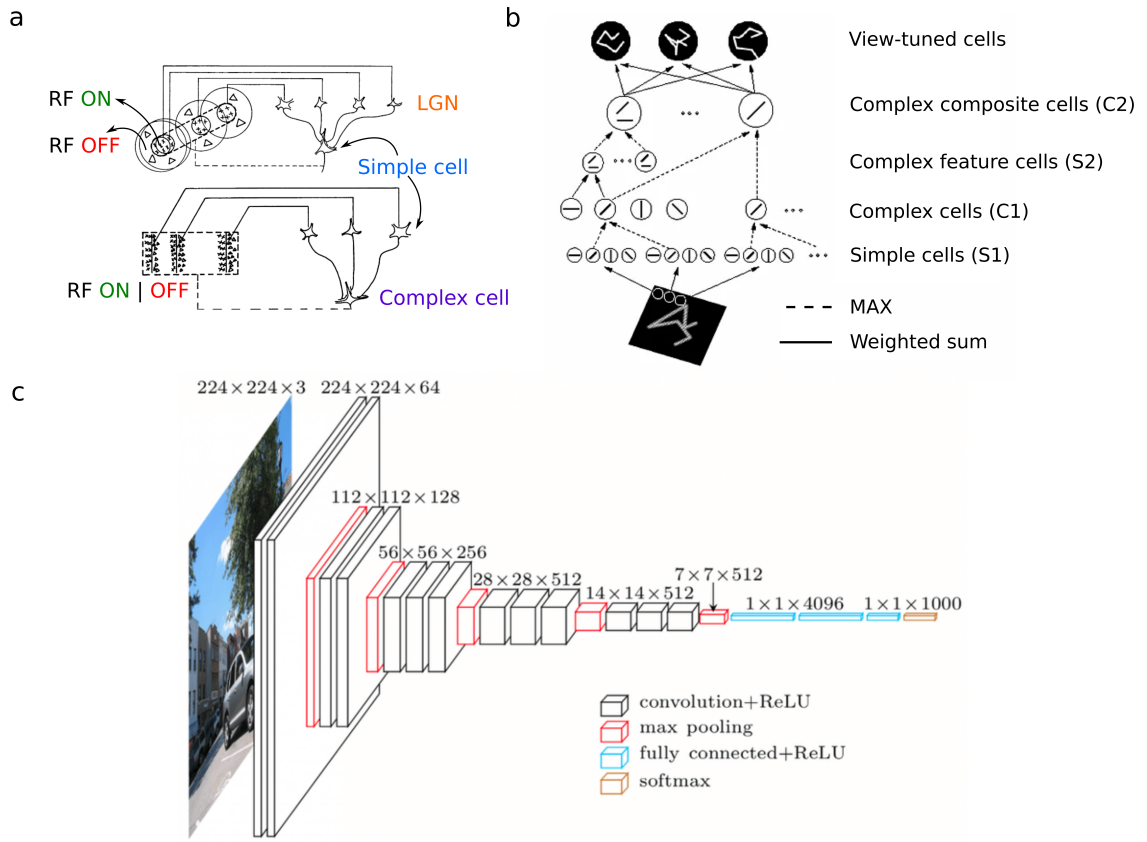


Figure 1.2: **Feedforward models of the ventral pathway.** (a) Hubel and Weisel's original hierarchical formulation that accounts for simple and complex cell responses to oriented bars in the cat striate cortex. Adopted and modified by author from [14] (b) The HMAX model extends Hubel and Weisel's proposal. The non-linear max pooling operator in this model helps reconcile physiological and behavioral observations, especially in the case of viewpoint and scale invariance, and has a plausible cortical circuitry implementation. Adopted from [45] (c) More recently, deep convolutional networks (DCNs) have cemented their status as the dominant models of feedforward processing and excel at several large-scale visual challenges. Depicted here is a popular neural architecture, VGG16. Adopted from [46].

and related models [49, 50] were an excellent start as models of shape perception, they did not explain any of the physiological properties of the ventral visual pathway. One of the first hierarchical models to do so was HMAX (Fig. 1.2b) [45]. HMAX extended the classic simple cells-complex cells framework to include the non-linear $\text{MAX}(\cdot)$ operation wherein a cell's activity was driven by the maximum value over all its afferents. This strategy was a specific implementation of pooling with biological plausibility that also accounted for experimentally observed IT responses to various input shapes. Later extensions of this model further improved its compatibility with newer anatomical and physiological findings, even accounting for behavioral patterns of errors in rapid visual categorization tasks [40].

The major insight that emerged from these early hierarchical models was that the composition of

linear-nonlinear transformations is a powerful computational strategy to build up general-purpose invariant properties. In what has proven a paradigm shift in the last decade, the emergence of large-scale image datasets and compute infrastructure has enabled “learning” the parameters of task-relevant transformations in a data-driven manner. In particular, a general class of hierarchical feedforward neural networks, known as deep convolutional networks (DCNs), have achieved remarkable feats of visual intelligence [51–55].

A prototypical DCN model (such as the one presented in Fig. 1.2c) consists of a series of transformations termed “layers”. The goal of each layer is to construct feature representations that increase in their level of abstraction and complexity along the hierarchy. Layers comprise a standard set of operations such as convolutions, pooling, normalization, and non-linear activations. DCNs utilize discrete convolutions as an efficient way to enforce spatially-local connectivity following the concept of receptive fields. These connectivity weights are referred to as “filters” or kernels. Once learned through the backpropagation algorithm [56], each filter responds strongly to specific spatially-local patterns of input. More importantly, each filter is replicated along the entire visual field, emulating the hypercolumnar organization of the visual cortex [57, 58]. Convolutions, by definition, are linear operations. To build representational power, a series of nonlinear transformations are subsequently applied to the convolutional outputs of neurons in a DCN. While these nonlinearities can have several different parameterizations, they usually consist of a pooling operation followed by an activation function whose outputs are then normalized. Pooling, as highlighted by the HMAX model among others, is an effective algorithm to aggregate over afferent inputs to a neuron. The post-synaptic activity of any given neuron in a DCN is computed as a non-linear transformation (such as a rectification, sigmoid, or a hyperbolic tangent function) of its aggregated convolutional afferents. As an additional trick to stabilize and speedup DCN training, post-synaptic activities are normalized using techniques such as batch [59], layer [60], or instance [61] normalization.

Though DCNs were primarily developed as computer vision algorithms, their impact on visual neuroscience has been significant. For the first time, researchers were able to optimize models for “performance” on large-scale naturalistic vision tasks such as image categorization [52, 53], scene classification [62], and object detection [63, 64] among others. Not only did these models rival human performance on challenging benchmarks [53, 65], they faithfully recapitulated primate behavioral patterns as well [66–68]. DCNs also provide computational level insights into the nature of visual representations and mechanisms supporting their transformations along the cortical hierarchy. Studies probing the internal representations of DCNs found the presence of Gabor-like kernels and color opponent neurons in the early layers and more sophisticated boundary, shape, and texture detectors as we go deeper. The emergence of these properties under nearly no constraints other than overall network performance confirmed its optimality for complex visual function as suggested by decades of electrophysiology research.

The internal representations of performance-optimized DCNs also happen to be highly predictive of neural responses in the ventral visual hierarchy [69, 70]. This success is another case in point in support of data-driven optimization. Earlier attempts to study neural encoding in higher visual areas

by fitting V4/IT neural tuning curves to image stimuli only resulted in limited predictive power [71]. The inability to robustly fit high-dimensional neural response patterns to high-dimensional image stimuli directly, however, is not surprising and can be explained simply by the curse of dimensionality. Optimizing networks for tasks, on the other hand, became feasible owing to large, curated image datasets. Interestingly, assaying a zoo of models revealed that a given model’s task-performance significantly correlated with the neural predictivity of its internal states [69]. This suggested, for the first time, that performance provides a rational method to a priori rule out neurally-inconsistent model architectures.

Though the focus in computer vision has predominantly been on this class of hierarchical feedforward ventral stream models, there have also been notable efforts in modeling dorsal computations [72–75]. Analogous to the impact that large-scale image datasets had on ventral stream models, the development of large-scale video datasets promoted the conceptualization of DCN models for motion processing [76, 77]. Particularly effective in this scenario was the adoption of the two stream hypothesis for DCNs [78]. The idea behind two-stream DCNs was to disentangle spatial and temporal processing by having two independent convolutional cascades, operating on different input features, that are then “fused” together in the latter stages. The spatial stream operated on raw pixel inputs and the temporal stream usually operated on optical flow maps. Modularizing DCNs in this manner proved useful for problems like action recognition as it promoted parameter efficiency, and as a consequence, sample efficiency. However, unlike the DCNs optimized for object recognition performance that explain ventral stream neural responses, there were no real parallels for dorsal stream neural explainability. Recently, Mineault et al. [75] showed that training DCNs to predict an agent’s self motion parameters in a simulated environment, as opposed to tasks such as video recognition, yielded significant fits to neural responses from dorsal stream areas such as MT and MST.

It is exceedingly clear that DCNs excel at approximating preattentive bottom-up processes and serve as functional abstractions of certain primate visual computations. Furthermore, mathematical theory suggests that DCNs are universal function approximators [79], that is, under certain assumptions, they would be able to learn any arbitrary input-output mapping. *Given that behavior is the ultimate output of the brain and that DCNs are theoretically capable of mapping any input stimulus to the appropriate behavioral output, would it be reasonable to suggest that they are, in their present form, adequate models of the visual system?*

The answer is a resounding “no”. The most obvious reason is that there is much more to visual cognition than pattern recognition. The functions supported by the visual system include relational and spatial reasoning, visual working memory, and imagery to name a few. All of these tasks require conceptual abstractions of the visual world beyond what is needed for recognition. Since DCNs are universal function approximators, would they not be able to succeed at these other tasks and provide insights into the computational mechanisms involved? While in theory they should, there are notable assumptions underlying the function approximation theory that are prohibitive. The universality of DCN’s is only valid in the limit of infinite network width and infinite data, both of which are not

practically feasible.

Inductive biases, or informed model design, can help alleviate some of this burden. One glaring mechanistic omission in DCNs is feedback circuitry. Though there have been preliminary attempts at incorporating feedback into DCNs (cf. skip connections in residual networks [53]), there are very limited computational-level hypotheses for how feedback impacts visual processing beyond the initial feedforward sweep. Physiological studies, however, have made progress in identifying the modulatory effect of feedback signals that are believed to impart computational flexibility in visual processing. In Section. 1.3 we will review some of this evidence and associated computational models before moving on to setup the research question from a neural dynamics perspective.

1.3 Contextual modulation of visual function

Visual processing is organized at multiple timescales [80]. As we saw, the bottom-up cascade of computations through the dorsal and ventral visual streams are already sufficient to implement sophisticated functions such as the recognition of objects presented in the central visual field. These computations are *fast* and only take on the order of $150ms$. However, in addition to these feedforward pathways, there are also horizontal, intra-area, and feedback projections in the visual cortex. Information flow through these “re-entrant” synapses operate on a *slower* timescale and are thought to support contextual guidance or feedback computations.

Some of the early evidence for feedback came from observing the dynamics underlying neural tuning properties [81]. For the longest time, area V1 was considered as a static bank of oriented filters. However, the temporal evolution of tuning following stimulus onset revealed that a neuron’s preferred orientation can change through time with varying degrees of sharpness [82]. Similar effects were also observed for color tuning [83]. More conclusive evidence came from face selective neurons in IT. While early in a trial, neural responses in these populations would only reliably predict if a presented stimulus was a face or not, they would start to express information about facial expressions and identities after longer latencies [84, 85]. Feedforward modeling alone was insufficient in explaining these dynamics shifts in population activity [86].

Feedback signaling, through recurrent pathways, reflected the modulation of a neuron’s response to stimuli presented in its classical receptive field (CRF) by information in the peripheral extra-classical receptive field (eCRF) [89]. This phenomenon is commonly referred to as *contextual modulation* to emphasize the point that neuronal responses are sensitive to context. Single neuron recordings from the primary visual cortex revealed the modulatory effect of stimuli extending beyond a neuron’s receptive field, specifically identifying that the strength of surround modulation depended on the match between center and surround stimuli [90, 91] (Fig. 1.3b). Subsequently, studies found that the suppression is, in fact, selective to those features that are shared between the center and the surround [88] (Fig. 1.3c). These physiological mechanisms also strongly correlated with psychophysical measures of the perceived perceptual properties of stimuli presented in the central visual field [92–94]. One such prominent effect is shown in Fig. 1.3a. Though the central ring is orange in both the left

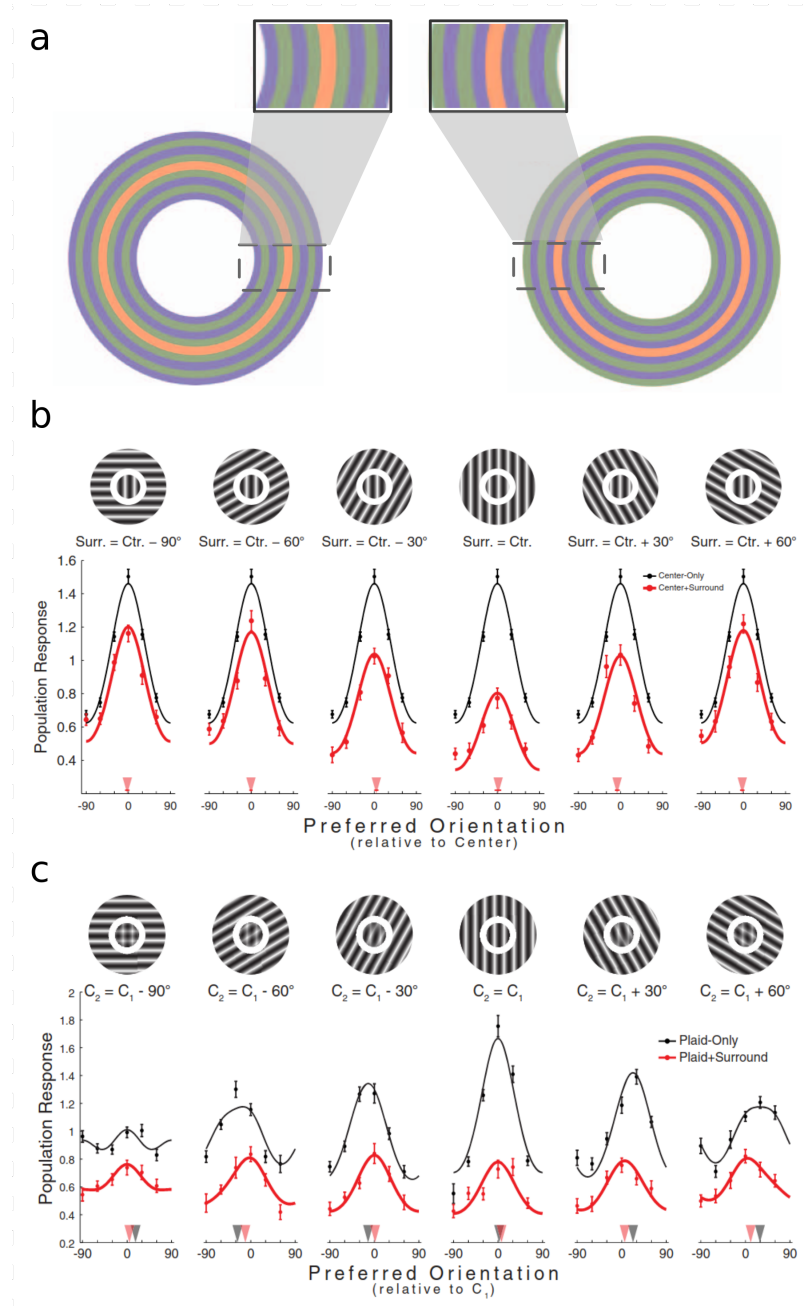


Figure 1.3: **Contextual modulation of behavioral and neural responses.** (a) The perceived color of the central orange ring is modulated by the color of the rings surrounding it. A purple surround strikingly shifts the perceived color of the central ring towards the pink end of the spectrum. Adopted and modified from [87]. (b) Multi-unit recordings from V1 revealed that surround suppression is dynamics and depended on the degree to which the center and surround stimuli match. Adopted from [88]. (c) In a more nuanced experiment, [88] find that surround suppression is, in fact, feature-specific and the suppression acts component-wise. Adopted from [88].

and right panels, its perceived color is shifted more toward the pink end of the color spectrum when its surrounded by the purple rings as opposed to the lime [87].

In addition to anatomical and physiological considerations, the effects of feedback are also apparent in primate behavior. Visual tasks that take primates more than $150ms$ to solve necessarily *have* to involve feedback computations. Longer latencies typically reflect the extent of “visual challenge” associated with the task. The notion of visual challenge is vague and can encompass a variety of computational demands. For example, consider the challenge of spotting your cat hiding beneath a pile of clothes. This is a visual search problem that can become arbitrarily hard based on the number of distracting objects in your visual field and the partial visibility of your cat. Computational resources, in this case, are devoted to suppress directing attention to distractor objects. Visual search studies have demonstrated that reaction times can increase by as much as $300ms$ based on task difficulty [95]. A separate computational challenge is that of pattern completion. While the primate visual system can recognize partially visible objects, neural selectivity in these cases only emerges later compared to selectivity for whole objects [86, 96] confirming the importance of feedback. The reaction times for recognizing partially occluded objects was also longer relative to recognition time for non-occluded objects [97]. Another computational challenge is the serial application of primitive visual operations [98]. The spread of attention in early visual cortex, for example, is theorized to operate by successively adding features to a growing object representation [99]. The more the number of serial computations, the more the computational challenge, and in turn the longer the response latency.

All of the aforementioned observations unequivocally establish the importance of feedback computations for visual function. Several computational models of feedback, specifically in the context of eCRF mechanisms, have been developed and primarily focused on two schools of thought: gain control and efficient coding [100]. Normalization models, in which the activity of a neuron is scaled by the responses of neighbouring neurons, explained several non-linear properties of V1 receptive fields [101]. Normalization supported adaptively maximizing sensitivity (gain control) and the winner-takes-all nature of this computation reduced redundancy and supported efficient coding. Predictive coding models suggested that the visual system can take advantage of statistical regularities of natural images and only use neural activity to encode deviations from regularities [102]. Finally, sparse coding theories used “sparseness” of a neuron’s responses as a proxy for its selectivity and postulated that sparse activations can be propagated efficiently through the visual system [103].

Though these models could explain physiological observations, they were unable to link said feedback mechanisms to human behavior. The recent development of large-scale computational models of feedback that achieved human-level performance on naturalistic vision tasks provided the first steps in this direction. It is worth noting that progress has been slower compared to the blistering advances in the deep feedforward regime, the primary reason being that tuning (or learning) the parameters of models with recurrent dynamics is much harder compared to their feedforward counterparts.

Mely et al. proposed one of the first models that unified several of the contextual phenomenon

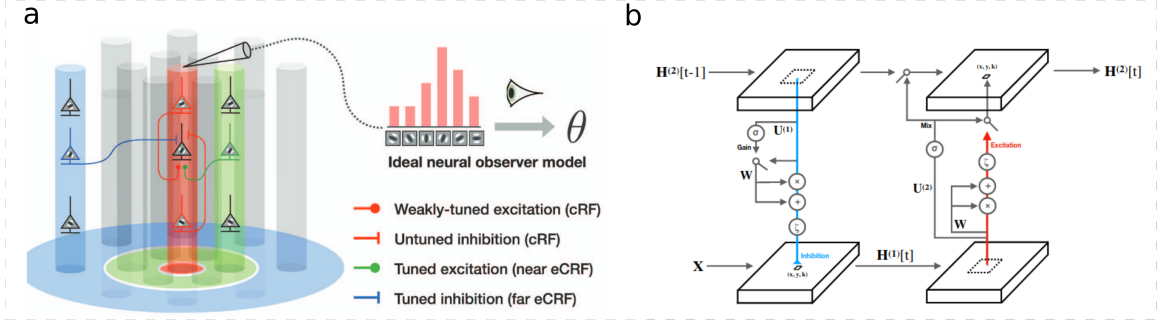


Figure 1.4: **Recurrent neural network models of primary visual cortex** (a) The proposed model of Mely et al. subscribed to a hyper-columnar architecture that distinguished near and far eCRF surrounds that had complementary effects on the CRF responses. The model reconciled a host of eCRF phenomenon and provided mechanistic insights into psychophysical measures of human behavior. Adopted from [87] (b) The horizontal gated recurrent unit (hGRU) from Linsley et al. [104] is a machine learning module that implements mechanisms described in [87]. The parameters of this unit, however, are learnable through gradient descent. Recurrent activities are denoted by H , kernels by W , input by X , and gates by U . $+$ and $\times$ symbolize linear and quadratic interactions respectively. Adopted from [104].

we discussed above while being consistent with psychophysical data [87]. The main feature of this model was the distinction between the near and far eCRFs that had complementary effects on the CRF response (Fig. 1.4a). While the near surround modulated the CRF response in a facilitatory manner, the far surround was suppressive. The model also featured a hypercolumnar organization with a bank of filters replicated across the visual field. The synaptic parameters of this model was constrained by anatomical data. This model was successful in reconciling several disparate contextual effects including tilt effects, color, motion, and depth induction, and color shifts from the perspective of CRF population dynamics.

While the model of Mely et al. was an impressive start, it had its limitations. Relying on sparse anatomical data to constrain the parameters of the recurrent circuits is both tedious and sub-optimal. The scale and scope of the model restricted its application to any realistic, large-scale visual problem, and thus was ultimately limited in determining if the motivating biological features of the model were consistent with human behavior. Towards resolving these issues, Linsley et al. proposed a canonical recurrent neural network (RNN) architecture that builds on this circuit model [104]. They constructed an RNN module by discretizing the neural field formulation of Mely et al. [87] and replacing time constants in the Euler update equations with learnable “gates”. The resulting horizontal gated recurrent unit (hGRU) was a modern machine learning formulation that could be trained in a purely data-driven manner with gradient descent to perform any desired task. Moreover, the hGRU was a lightweight “layer” that could be stacked on top of any feature extractor to simulate appropriate recurrent dynamics. The hGRU outperformed several state-of-the-art deep feedforward networks, with orders of magnitudes fewer parameters, on a perceptual grouping task that relied on eCRF mechanisms. In a subsequent study, Kim et al. extended the hGRU formulation to a stacked deep recurrent network that in addition to lateral feedback also had mechanisms for top-down

feedback [105].

Though physiologically-plausible deep neural network models of feedback are few and far between, several other functional approximations have proved to be immensely helpful. A notable example is the U-Net [106]. U-Nets employ an encoder-decoder style architecture but include lateral feedforward connections between encoding and decoding layers at each level. These feedforward connections are the primary source of feedback in this model. Similar to the U-Nets are the Feature Pyramid Networks (FPNs) [64]. FPNs are multi-scale, top-down architectures that provide feedback from high-level semantic features down to the local receptive fields. Both the U-Net and FPNs have become canonical backbone architectures that consistently outperform feedforward networks in tasks such as image segmentation and object detection. Capsule networks [107] are a third different model class that functionally implements feedback via lateral competition between units driven by top-down inputs.

In all of the aforementioned computational and functional models of feedback, network *architecture* was the key factor in determining the capabilities of the system. However, for recurrent networks, which is the case for feedback circuits, network architecture is *not* the only factor determining the neural population dynamics that ultimately specify the algorithms executed by the circuits. Understanding the relationship between network connectivity and activity dynamics has been a long-standing goal in computational neuroscience. Recently, there has been a push towards merging the idea of function approximation with system identification. Stated differently, can we (in a black-box manner) identify the rules that govern the temporality of dynamical features in neural population responses? In Section. 1.4 we will briefly review advances in this regard.

1.4 Computation through dynamics

Single-neuron electrophysiology and computational models have led to several insights pertaining to the selectivity and invariance properties of neural responses to stimuli. HMAX and modern DCNs are amongst this class of feedforward models. However, a paradigm shift in computational neuroscience was the observation that global population responses contained useful semantic representations of sensory and cognitive processing that were not apparent at a single neuron level [108]. It has now become common wisdom that neural population dynamics reflect the inner workings of complex systems and helps shed light on the algorithms they help implement [109, 110]. Specifically, there were two parallel lines of investigation that provided insights in this regard; one that focused on the emergence of patterns in neural population activity from network specification and the other that focused on interpreting these patterns and their computational benefits.

Cortical dynamics are constituted by both the activities driven by a stimulus or an external input and activities that are internally generated and sustained [111, 112]. Feedforward models simply cannot account for the latter. This particular deficiency highlights the importance of recurrent neural networks models in the context of modeling cortical function. Among the first efforts in relating the structure of a network of excitatory and inhibitory neurons to activity dynamics was the mean field

approach of Wilson and Cowan [113, 114]. Within this framework, a set of governing rules, termed the *neural field* equations, were used to specify the evolution of a network’s global population firing rate. The goal of several subsequent studies was then to study the link between the parameters of neural field equations and the macroscopic properties of a network [115, 116].

Though we still lack a theoretical framework that causally links *every* network’s architecture with its dynamics, there were certain special model families for which this was possible. One such extensively studied model class was the (fully) randomly connected recurrent network [117, 118]. These networks were able to internally sustain irregular and oscillatory activity patterns and closely resembled cortical dynamics [119]. However, they were not efficient in terms of information encoding given their full-rank structure [120], had limited computational versatility, and only demonstrated minimal variability in their responses to external stimuli [121]. On the other end of the spectrum were recurrent networks with highly structured connectivity. These networks usually incorporated both special connectivity patterns such as block or low-rank matrices and hard-wired neural tuning properties [122, 123]. In general, the practice of engineering connectivity patterns to construct networks that had specific computational abilities was widespread at the time. Perhaps the most representative model in this regard was the Hopfield network [124]. In a Hopfield network, neurons had a binary state value and were symmetrically connected to their neighbours. These connectivity weights were subsequently updated based on the Hebbian plasticity rule [125]. The symmetric weight constraint provided some nice theoretical guarantees for the system to function as a content-addressable memory module. Similarly, synfire chains, in which groups of neurons are coupled in a feedforward manner, were constructs that supported the exhibition of neural synchrony [126].

Towards interpreting patterns in neural population activity and their related computational benefits, models have recently started to embrace a dynamical systems perspective [127]. In physics, a dynamical system is essentially a particle or group of particles whose state *predictably* varies over time. In neuroscience, neurons are the “particles” that constitute the dynamical system and their firing rates at any given time is considered the “state” of the system. Consider a network of N neurons. Their firing rates at any given time t can be expressed as the vector $\mathbf{x}(t) = [x_1(t), x_2(t), \dots, x_N(t)]$. In its most general form, the resulting dynamical system can be mathematically written as follows:

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), \mathbf{u}(t), t),$$

where $\mathbf{u}(t)$ describes any external input at a given point in time and $f(\cdot)$ is the function specifying the temporal evolution of $\mathbf{x}(t)$. A toy illustration of this perspective is shown in Fig. 1.5. The general idea behind such a formulation is that the temporal evolution of a population’s activity ultimately results in the neural population carrying out specific functions [128, 129].

The dynamical systems perspective started to gain traction owing to its conceptual tightness, simplicity, and ability to account for various interesting phenomena. The choice of the function f was clearly an important design choice and it depended on one of two problem formulations. In one case, the modeler has access to large-scale neural recordings, $\mathbf{x}$, and wants to find an appropriate f based on these patterns. In the other case, the modeler has access to temporal input-output relationships

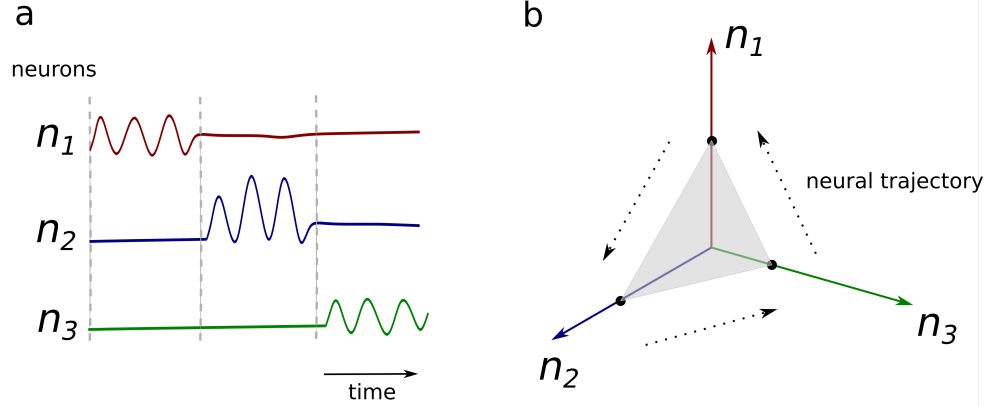


Figure 1.5: **A toy illustration of neural population “states” and dynamics.** (a) Consider a population consisting of three neurons. In any given interval of time (vertical, gray dashed lines), individual neurons can have varying levels of activity. By summarizing the overall activity of each neuron in a time window (either by binning and counting spikes or by taking the average firing rate), one can effectively represent the “state” of this population by a vector in three dimensions, with one coordinate per neuron. (b) Tracking the evolution of this three dimensional vector through time (i.e., through successive time windows) results in a path that is called the neural trajectory. The gray shaded region denotes the subspace of $\mathbb{R}^3$ in which the activity patterns from this example reside.

while the system as a whole is performing a task, and the challenge is in identifying the right f that successfully transforms inputs into appropriate outputs. In either case, we shall refer to f as the “model”.

Some of the hand-engineered model architectures we reviewed before, such as synfire chains, are excellent examples of the former, i.e., constructing a model that can explain a specific neural phenomenon. In the case of synfire chains, the desired temporal property was the emergence of volleys of synchronized neural activity. The downside of this manual construction process was that it was excessively tedious and sub-optimal. Recent advances in machine learning, especially numerical optimization techniques, have started to play a huge role in circumventing this problem. Nowadays, a common strategy is to parametrically specify f as f_θ and learn the parameters θ in a data-driven manner. Effectively, θ describes the architecture and connectivity patterns of model f_θ . Machine learning approaches were also increasingly successful in training dynamical systems on input-output relationships while carrying out specific tasks.

One particularly successful strategy involved removing non-linearities in the function f_θ and instead modeling it as a Linear Dynamical System (LDS). In this case, one can re-write the canonical formulation as follows:

$$\frac{d\mathbf{x}(t)}{dt} = f_\theta(\mathbf{x}(t), \mathbf{u}(t)) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t),$$

where $\theta := [\mathbf{A}, \mathbf{B}]$ are the learned parameters. Linear models were easy to train and analyze, and could reconcile a few different behaviors including oscillations and decaying dynamics. In fact, [130] argued that this approach can even benefit complex non-linear dynamical systems after decomposing

them into linear systems at certain points in their dynamical landscape. Modeling f_θ as a small black-box neural network that can subsequently be “linearized” and analyzed provided a trade-off between interpretability and expressivity. This has been particularly effective in capturing local dynamics in the motor cortex as well as behavioral effects such as reaching and grasping reaction times [129, 131–133].

In parallel to these developments, the machine learning community has also invested effort towards developing solutions for learning differential equations directly from data [134, 135]. Essentially these methods aimed to approximate the function f_θ with deep neural networks, in contrast to the simpler, more interpretable RNNs used in the computational neuroscience literature. Amongst the most popular approaches in this domain are the NeuralODEs [136], its derivatives [137–140], and the DeepONet [141]. These models have performed incredibly well on simple time-series classification, few-shot generalization, and image generation tasks. We will compare and contrast some of these approaches in subsequent chapters of this thesis, but for now, we note that the general goal of these model families is fundamentally different from the computational neuroscience approaches. The deep dynamical system methods look to leverage properties of continuous and differentiable flows to provide efficient alternatives to very large feedforward networks, which in turn provides computational and sample efficiency. They *do not* share the viewpoint that dynamical motifs are the fundamental currency in a computational space. Moreover, most of the black-box approximations (of f_θ) are carried out by feedforward or quasi-feedforward network architectures and do not focus on providing computational or behavioral insights into the temporal evolution of neural population dynamics.

1.5 Goal of this thesis

Our introductory remarks and review of existing literature was aimed to highlight the desiderata for this thesis. We saw that complex visual function carried out by biological systems necessarily involve feedback mechanisms and so should a model hoping to emulate such capabilities efficiently. Moreover, large-scale naturalistic visual challenge benchmarks are still dominated by deep feedforward networks despite general consensus about their limitations. We also saw that feedback circuits generate complex patterns of neural activity, and, if one has a handle on that, they can be orchestrated to help implement desired computations. However, these demonstrations have been limited to relatively simple model classes. And finally, the link between activity dynamics in feedback models of complex visual function and network behavior is still unspecified.

The goal of this thesis is to integrate the dynamical systems perspective from computational neuroscience with large-scale deep recurrent neural networks (RNNs) that are feedback systems capable of performing complex visual function. First, we study the algorithmic and practical benefits of controlling the neural dynamics of biologically-inspired, large-scale RNNs. Second, we observe that doing so imparts several computational

affordances for visual function and provides a framework to study the interactions between vision and higher cognitive functions, like decision-making and mental imagery.

The thesis is subsequently divided into four chapters, with the contributions of each as follows.

- **Chapter 2** reviews foundational mathematical concepts and definitions in the theory of differential equations and dynamical systems. We review techniques to analyze and quantify the stability of solutions and outline canonical dynamical patterns.
- **Chapter 3** outlines the primary bottleneck that prevents training RNNs at scale on large-scale visual benchmarks. We propose and implement an effective solution for this problem that is inspired by attractor dynamics. We analyze the computational superiority of our RNNs and observe the mechanisms that support their behavior.
- **Chapter 4** explores properties of our RNN model family that emerge from task-optimization. We demonstrate that the visual representations emerging from our model dynamics can ably support downstream utilization for decision-making. We also highlight how our models can potentially line up with human behavior in two separate use cases.
- In **Chapter 5** we move from modeling activity dynamics to modeling synaptic dynamics and briefly make the argument that fast-changing synaptic properties can support attractor dynamics at the population level. Moreover, we propose a novel framework to study how learned complex visual representations are maintained and manipulated in feedback networks.

1.6 References

- [1] Daniel J Felleman and David C Van Essen. “Distributed hierarchical processing in the primate cerebral cortex.” In: *Cerebral cortex (New York, NY: 1991)* 1.1 (1991), pp. 1–47.
- [2] Kathleen S Rockland and Deepak N Pandya. “Laminar origins and terminations of cortical connections of the occipital lobe in the rhesus monkey”. In: *Brain research* 179.1 (1979), pp. 3–20.
- [3] David C Van Essen and John HR Maunsell. “Hierarchical organization and functional streams in the visual cortex”. In: *Trends in neurosciences* 6 (1983), pp. 370–375.
- [4] Jean Bullier. “Cortical connections and functional interactions between visual cortical areas”. In: *The Neuropsychology of Vision*. Oxford University Press, Oct. 2003. ISBN: 9780198505822. DOI: 10.1093/acprof:oso/9780198505822.003.0002. eprint: https://academic.oup.com/book/0/chapter/145194400/chapter-ag-pdf/45009729/book_3768_section_145194400.ag.pdf. URL: <https://doi.org/10.1093/acprof:oso/9780198505822.003.0002>.
- [5] Jean Bullier. “Integrated model of visual processing”. In: *Brain research reviews* 36.2-3 (2001), pp. 96–107.
- [6] Paul-Antoine Salin and Jean Bullier. “Corticocortical connections in the visual system: structure and function”. In: *Physiological reviews* 75.1 (1995), pp. 107–154.
- [7] Shabtai Barash et al. “Saccade-related activity in the lateral intraparietal area. II. Spatial properties”. In: *Journal of neurophysiology* 66.3 (1991), pp. 1109–1124.
- [8] Gordon C Baylis, Edmund T Rolls, and CM Leonard. “Functional subdivisions of the temporal lobe neocortex”. In: *Journal of Neuroscience* 7.2 (1987), pp. 330–342.
- [9] MH Munk et al. “Visual latencies in cytochrome oxidase bands of macaque area V2.” In: *Proceedings of the National Academy of Sciences* 92.4 (1995), pp. 988–992.
- [10] Shintaro Funahashi, Charles J Bruce, and Patricia S Goldman-Rakic. “Visuospatial coding in primate prefrontal neurons revealed by oculomotor paradigms”. In: *Journal of neurophysiology* 63.4 (1990), pp. 814–831.
- [11] K Kawano et al. “Neural activity in cortical area MST of alert monkey during ocular following responses”. In: *Journal of neurophysiology* 71.6 (1994), pp. 2305–2324.
- [12] John H Maunsell and Jay R Gibson. “Visual response latencies in striate cortex of the macaque monkey”. In: *journal of Neurophysiology* 68.4 (1992), pp. 1332–1344.
- [13] Lionel G Nowak and Jean Bullier. “The timing of information transfer in the visual system”. In: *Extrastriate cortex in primates*. Springer, 1997, pp. 205–241.
- [14] David H Hubel and Torsten N Wiesel. “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex”. In: *The Journal of physiology* 160.1 (1962), p. 106.

- [15] David H Hubel and Torsten N Wiesel. “Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat”. In: *Journal of neurophysiology* 28.2 (1965), pp. 229–289.
- [16] H Sherk and M Ombrellaro. “The retinotopic match between area 17 and its targets in visual suprasylvian cortex”. In: *Experimental brain research* 72.2 (1988), pp. 225–236.
- [17] David J Price et al. “Functional organization of corticocortical projections from area 17 to area 18 in the cat’s visual cortex”. In: *Journal of Neuroscience* 14.5 (1994), pp. 2732–2746.
- [18] KEIJI Tanaka. “Cross-correlation analysis of geniculostriate neuronal relationships in cats”. In: *Journal of neurophysiology* 49.6 (1983), pp. 1303–1318.
- [19] Stephen A Engel, Gary H Glover, and Brian A Wandell. “Retinotopic organization in human visual cortex and the spatial precision of functional MRI.” In: *Cerebral cortex (New York, NY: 1991)* 7.2 (1997), pp. 181–192.
- [20] Gavin Perry et al. “Retinotopic mapping of the primary visual cortex—a challenge for MEG imaging of the human cortex”. In: *European Journal of Neuroscience* 34.4 (2011), pp. 652–661.
- [21] Russell L DeValois and Karen K De Valois. *Spatial vision*. 14. Oxford University Press on Demand, 1990.
- [22] Karl R Gegenfurtner, Daniel C Kiper, et al. “Color vision”. In: *Annual review of neuroscience* 26.1 (2003), pp. 181–206.
- [23] Gregory D Horwitz and Charles A Hass. “Nonlinear analysis of macaque V1 color tuning reveals cardinal directions for cortical color processing”. In: *Nature neuroscience* 15.6 (2012), pp. 913–919.
- [24] Rüdiger Von der Heydt, Esther Peterhans, and Gunter Baumgartner. “Illusory contours and cortical neuron responses”. In: *Science* 224.4654 (1984), pp. 1260–1262.
- [25] Akiyuki Anzai, Xinmiao Peng, and David C Van Essen. “Neurons in monkey visual area V2 encode combinations of orientations”. In: *Nature neuroscience* 10.10 (2007), pp. 1313–1321.
- [26] Rüdiger Von Der Heydt, Hong Zhou, and Howard S Friedman. “Representation of stereoscopic edges in monkey visual cortex”. In: *Vision research* 40.15 (2000), pp. 1955–1967.
- [27] Anitha Pasupathy and Charles E Connor. “Responses to contour features in macaque area V4”. In: *Journal of neurophysiology* 82.5 (1999), pp. 2490–2502.
- [28] Anitha Pasupathy and Charles E Connor. “Shape representation in area V4: position-specific tuning for boundary conformation”. In: *Journal of neurophysiology* (2001).
- [29] Charles G Gross, CE de Rocha-Miranda, and DB Bender. “Visual properties of neurons in inferotemporal cortex of the Macaque.” In: *Journal of neurophysiology* 35.1 (1972), pp. 96–111.
- [30] Charles G Gross. “Representation of visual stimuli in inferior temporal cortex”. In: *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 335.1273 (1992), pp. 3–10.

- [31] John H Maunsell and David C Van Essen. “Functional properties of neurons in middle temporal visual area of the macaque monkey. I. Selectivity for stimulus direction, speed, and orientation”. In: *Journal of neurophysiology* 49.5 (1983), pp. 1127–1147.
- [32] Rf Dubner and SM Zeki. “Response properites and receptive fields of cells in an anatomically defined region of the superior temporal sulcus in the monkey.” In: *Brain research* (1971).
- [33] Walter Pohl. “Dissociation of spatial discrimination deficits following frontal and parietal lesions in monkeys.” In: *Journal of comparative and physiological psychology* 82.2 (1973), p. 227.
- [34] Mortimer Mishkin, Leslie G Ungerleider, and Kathleen A Macko. “Object vision and spatial vision: two cortical pathways”. In: *Trends in neurosciences* 6 (1983), pp. 414–417.
- [35] Keiji Tanaka et al. “Inferotemporal cortex and object vision”. In: *Annual review of neuroscience* 19.1 (1996), pp. 109–139.
- [36] John H Maunsell and David C Van Essen. “Functional properties of neurons in middle temporal visual area of the macaque monkey. II. Binocular interactions and sensitivity to binocular disparity”. In: *Journal of neurophysiology* 49.5 (1983), pp. 1148–1167.
- [37] Semir M Zeki. “Functional organization of a visual area in the posterior bank of the superior temporal sulcus of the rhesus monkey”. In: *The Journal of physiology* 236.3 (1974), pp. 549–573.
- [38] Yves Rossetti. “Implicit perception in action: Short-lived motor representation of space.” In: (2001).
- [39] Simon Thorpe, Denis Fize, and Catherine Marlot. “Speed of processing in the human visual system”. In: *nature* 381.6582 (1996), pp. 520–522.
- [40] Thomas Serre, Aude Oliva, and Tomaso Poggio. “A feedforward architecture accounts for rapid categorization”. In: *Proceedings of the national academy of sciences* 104.15 (2007), pp. 6424–6429.
- [41] Rufin VanRullen. “The power of the feed-forward sweep”. In: *Advances in Cognitive Psychology* 3.1-2 (2007), p. 167.
- [42] Chou P Hung et al. “Fast readout of object identity from macaque inferior temporal cortex”. In: *Science* 310.5749 (2005), pp. 863–866.
- [43] Hesheng Liu et al. “Timing, timing, timing: fast decoding of object information from intracranial field potentials in human visual cortex”. In: *Neuron* 62.2 (2009), pp. 281–290.
- [44] JH Kaas and LA Krubitzer. “The organization of extrastriate visual cortex”. In: *Neuroanatomy of Visual Pathways and their Retinotopic Organization* 3 (1991), pp. 302–359.
- [45] Maximilian Riesenhuber and Tomaso Poggio. “Hierarchical models of object recognition in cortex”. In: *Nature neuroscience* 2.11 (1999), pp. 1019–1025.

- [46] Manolis Loukidakis, José Cano, and Michael O’Boyle. “Accelerating deep neural networks on low power heterogeneous architectures”. In: (2018).
- [47] David Marr. *Vision: A computational investigation into the human representation and processing of visual information*. MIT press, 2010.
- [48] Kunihiko Fukushima and Sei Miyake. “Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition”. In: *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.
- [49] David I Perrett and Mike W Oram. “Neurophysiology of shape processing”. In: *Image and Vision Computing* 11.6 (1993), pp. 317–333.
- [50] Edmund T Rolls and T Milward. “A model of invariant object recognition in the visual system: learning rules, activation functions, lateral inhibition, and information-based performance measures”. In: *Neural computation* 12.11 (2000), pp. 2547–2572.
- [51] Le Cun Yan, B Yoshua, and H Geoffrey. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.
- [52] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.
- [53] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [54] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.
- [55] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).
- [56] Raul Rojas. “The backpropagation algorithm”. In: *Neural networks*. Springer, 1996, pp. 149–182.
- [57] Rodrigo O Kuljis and Pasko Rakic. “Hypercolumns in primate visual cortex can develop in the absence of cues from photoreceptors.” In: *Proceedings of the National Academy of Sciences* 87.14 (1990), pp. 5303–5306.
- [58] Margaret S Livingstone and David H Hubel. “Anatomy and physiology of a color system in the primate visual cortex”. In: *Journal of Neuroscience* 4.1 (1984), pp. 309–356.
- [59] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.
- [60] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. “Layer normalization”. In: *arXiv preprint arXiv:1607.06450* (2016).
- [61] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. “Instance normalization: The missing ingredient for fast stylization”. In: *arXiv preprint arXiv:1607.08022* (2016).

- [62] Delu Zeng et al. “Deep learning for scene classification: A survey”. In: *arXiv preprint arXiv:2101.10531* (2021).
- [63] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.
- [64] Tsung-Yi Lin et al. “Feature pyramid networks for object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.
- [65] Ira Kemelmacher-Shlizerman et al. “The megaface benchmark: 1 million faces for recognition at scale”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4873–4882.
- [66] Rishi Rajalingham, Kailyn Schmidt, and James J DiCarlo. “Comparison of object recognition behavior in human and monkey”. In: *Journal of Neuroscience* 35.35 (2015), pp. 12127–12136.
- [67] Saeed Reza Kheradpisheh et al. “Deep networks can resemble human feed-forward vision in invariant object recognition”. In: *Scientific reports* 6.1 (2016), pp. 1–24.
- [68] Sven Eberhardt, Jonah G Cader, and Thomas Serre. “How deep is the feature analysis underlying rapid visual categorization?” In: *Advances in neural information processing systems* 29 (2016).
- [69] Daniel LK Yamins et al. “Performance-optimized hierarchical models predict neural responses in higher visual cortex”. In: *Proceedings of the national academy of sciences* 111.23 (2014), pp. 8619–8624.
- [70] Martin Schrimpf et al. “Brain-score: Which artificial neural network for object recognition is most brain-like?” In: *BioRxiv* (2020), p. 407007.
- [71] Jack L Gallant et al. “Neural responses to polar, hyperbolic, and Cartesian gratings in area V4 of the macaque monkey”. In: *Journal of neurophysiology* 76.4 (1996), pp. 2718–2739.
- [72] Martin A Giese and Tomaso Poggio. “Neural mechanisms for the recognition of biological movements”. In: *Nature Reviews Neuroscience* 4.3 (2003), pp. 179–192.
- [73] Karen Simonyan and Andrew Zisserman. “Two-stream convolutional networks for action recognition in videos”. In: *Advances in neural information processing systems* 27 (2014).
- [74] Yuxuan Zhao et al. “Improved two-stream model for human action recognition”. In: *EURASIP Journal on Image and Video Processing* 2020.1 (2020), pp. 1–9.
- [75] Patrick Mineault et al. “Your head is there to move you around: Goal-driven models of the primate dorsal pathway”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 28757–28771.
- [76] Hildegard Kuehne et al. “HMDB: a large video database for human motion recognition”. In: *2011 International conference on computer vision*. IEEE. 2011, pp. 2556–2563.
- [77] Will Kay et al. “The kinetics human action video dataset”. In: *arXiv preprint arXiv:1705.06950* (2017).

- [78] Melvyn A Goodale and A David Milner. “Separate visual pathways for perception and action”. In: *Trends in neurosciences* 15.1 (1992), pp. 20–25.
- [79] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.
- [80] Victor AF Lamme and Pieter R Roelfsema. “The distinct modes of vision offered by feedforward and recurrent processing”. In: *Trends in neurosciences* 23.11 (2000), pp. 571–579.
- [81] Charles D Gilbert and Torsten N Wiesel. “Receptive field dynamics in adult primary visual cortex”. In: *Nature* 356.6365 (1992), pp. 150–152.
- [82] Dario L Ringach, Michael J Hawken, and Robert Shapley. “Dynamics of orientation tuning in macaque primary visual cortex”. In: *Nature* 387.6630 (1997), pp. 281–284.
- [83] Nicolas P Cottaris and Russell L De Valois. “Temporal dynamics of chromatic tuning in macaque primary visual cortex”. In: *Nature* 395.6705 (1998), pp. 896–900.
- [84] Martin J Tovée. “Neuronal Processing: How fast is the speed of thought?” In: *Current Biology* 4.12 (1994), pp. 1125–1127.
- [85] Yasuko Sugase et al. “Global and fine information coded by single neurons in the temporal visual cortex”. In: *Nature* 400.6747 (1999), pp. 869–873.
- [86] Hanlin Tang et al. “Spatiotemporal dynamics underlying object completion in human ventral visual cortex”. In: *Neuron* 83.3 (2014), pp. 736–748.
- [87] David A Mély, Drew Linsley, and Thomas Serre. “Complementary surrounds explain diverse contextual phenomena across visual modalities.” In: *Psychological review* 125.5 (2018), p. 769.
- [88] Alexander R Trott and Richard T Born. “Input-gain control produces feature-specific surround suppression”. In: *Journal of Neuroscience* 35.12 (2015), pp. 4973–4982.
- [89] John Allman, Francis Miezin, and EveLynn McGuinness. “Stimulus specific responses from beyond the classical receptive field: neurophysiological mechanisms for local-global comparisons in visual neurons”. In: *Annual review of neuroscience* 8.1 (1985), pp. 407–430.
- [90] Adam M Sillito et al. “Visual cortical mechanisms detecting focal orientation discontinuities”. In: *Nature* 378.6556 (1995), pp. 492–496.
- [91] James R Cavanaugh, Wyeth Bair, and J Anthony Movshon. “Selectivity and spatial distribution of signals from the receptive field surround in macaque V1 neurons”. In: *Journal of neurophysiology* (2002).
- [92] James J Knierim and David C Van Essen. “Neuronal responses to static texture patterns in area V1 of the alert macaque monkey”. In: *Journal of neurophysiology* 67.4 (1992), pp. 961–980.
- [93] Andrew F Rossi, Cynthia D Rittenhouse, and Michael A Paradiso. “The representation of brightness in primary visual cortex”. In: *Science* 273.5278 (1996), pp. 1104–1107.

- [94] T Wachtler, TJ Sejnowski, and TD Albright. “Interactions between stimulus and background chromaticities in macaque primary visual cortex”. In: *Investigative Ophthalmology & Visual Science*. Vol. 40. 4. 1999, S641–S641.
- [95] Jeremy M Wolfe. “What can 1 million trials tell us about visual search?” In: *Psychological Science* 9.1 (1998), pp. 33–39.
- [96] Yoshito Kosai et al. “The role of visual area V4 in the discrimination of partially occluded shapes”. In: *Journal of Neuroscience* 34.25 (2014), pp. 8570–8584.
- [97] Richard F Murray, Allison B Sekuler, and Patrick J Bennett. “Time course of amodal completion revealed by a shape discrimination task”. In: *Psychonomic Bulletin & Review* 8.4 (2001), pp. 713–720.
- [98] Shimon Ullman. “Visual routines”. In: *Readings in computer vision*. Elsevier, 1987, pp. 298–328.
- [99] Arezoo Pooresmaeili and Pieter R Roelfsema. “A growth-cone model for the spread of object-based attention during contour grouping”. In: *Current Biology* 24.24 (2014), pp. 2869–2877.
- [100] Alessandra Angelucci and S Shushruth. “Beyond the classical receptive field: surround modulation in primary visual cortex”. In: *The new visual neurosciences* (2013), pp. 425–444.
- [101] Matteo Carandini and David J Heeger. “Normalization as a canonical neural computation”. In: *Nature Reviews Neuroscience* 13.1 (2012), pp. 51–62.
- [102] Rajesh PN Rao and Dana H Ballard. “Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects”. In: *Nature neuroscience* 2.1 (1999), pp. 79–87.
- [103] Bruno A Olshausen and David J Field. “Sparse coding with an overcomplete basis set: A strategy employed by V1?” In: *Vision research* 37.23 (1997), pp. 3311–3325.
- [104] Drew Linsley et al. “Learning long-range spatial dependencies with horizontal gated recurrent units”. In: *Advances in neural information processing systems* 31 (2018).
- [105] Junkyung Kim et al. “Disentangling neural mechanisms for perceptual grouping”. In: *arXiv preprint arXiv:1906.01558* (2019).
- [106] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.
- [107] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. “Dynamic routing between capsules”. In: *Advances in neural information processing systems* 30 (2017).
- [108] Apostolos P Georgopoulos, Andrew B Schwartz, and Ronald E Kettner. “Neuronal population coding of movement direction”. In: *Science* 233.4771 (1986), pp. 1416–1419.
- [109] Shreya Saxena and John P Cunningham. “Towards the neural population doctrine”. In: *Current opinion in neurobiology* 55 (2019), pp. 103–111.

- [110] Mikail Khona and Ila R Fiete. “Attractor and integrator networks in the brain”. In: *arXiv preprint arXiv:2112.03978* (2021).
- [111] Albert Compte et al. “Synaptic mechanisms and network dynamics underlying spatial working memory in a cortical network model”. In: *Cerebral cortex* 10.9 (2000), pp. 910–923.
- [112] Albert Compte et al. “Temporally irregular mnemonic persistent activity in prefrontal neurons of monkeys during a delayed response task”. In: *Journal of neurophysiology* 90.5 (2003), pp. 3441–3454.
- [113] Hugh R Wilson and Jack D Cowan. “Excitatory and inhibitory interactions in localized populations of model neurons”. In: *Biophysical journal* 12.1 (1972), pp. 1–24.
- [114] Hugh R Wilson and Jack D Cowan. “A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue”. In: *Kybernetik* 13.2 (1973), pp. 55–80.
- [115] Jack D Cowan, Jeremy Neuman, and Wim van Drongelen. “Wilson–Cowan equations for neocortical dynamics”. In: *The Journal of Mathematical Neuroscience* 6.1 (2016), pp. 1–24.
- [116] Hugh R Wilson and Jack D Cowan. “Evolution of the Wilson–Cowan equations”. In: *Biological cybernetics* 115.6 (2021), pp. 643–653.
- [117] Haim Sompolinsky, Andrea Crisanti, and Hans-Jurgen Sommers. “Chaos in random neural networks”. In: *Physical review letters* 61.3 (1988), p. 259.
- [118] Carl Van Vreeswijk and Haim Sompolinsky. “Chaos in neuronal networks with balanced excitatory and inhibitory activity”. In: *Science* 274.5293 (1996), pp. 1724–1726.
- [119] Michael N Shadlen and William T Newsome. “The variable discharge of cortical neurons: implications for connectivity, computation, and information coding”. In: *Journal of neuroscience* 18.10 (1998), pp. 3870–3896.
- [120] Ryan C Williamson et al. “Scaling properties of dimensionality reduction for neural populations and network models”. In: *PLoS computational biology* 12.12 (2016), e1005141.
- [121] Kanaka Rajan and Larry F Abbott. “Eigenvalue spectra of random matrices for neural networks”. In: *Physical review letters* 97.18 (2006), p. 188104.
- [122] Daniel J Amit and Nicolas Brunel. “Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex.” In: *Cerebral cortex (New York, NY: 1991)* 7.3 (1997), pp. 237–252.
- [123] Francesca Mastrogiuseppe and Srdjan Ostojic. “Linking connectivity, dynamics, and computations in low-rank recurrent neural networks”. In: *Neuron* 99.3 (2018), pp. 609–623.
- [124] John J Hopfield. “Neural networks and physical systems with emergent collective computational abilities.” In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.
- [125] Terrence J Sejnowski and Gerald Tesauro. “The Hebb rule for synaptic plasticity: algorithms and implementations”. In: *Neural models of plasticity*. Elsevier, 1989, pp. 94–103.

- [126] Moshe Abeles. *Corticonics: Neural circuits of the cerebral cortex*. Cambridge University Press, 1991.
- [127] Steven H Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.
- [128] Chethan Pandarinath et al. “Latent factors and dynamics in motor cortex and their application to brain-machine interfaces”. In: *Journal of Neuroscience* 38.44 (2018), pp. 9390–9401.
- [129] Chethan Pandarinath et al. “Inferring single-trial neural population dynamics using sequential auto-encoders”. In: *Nature methods* 15.10 (2018), pp. 805–815.
- [130] David Sussillo and Omri Barak. “Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks”. In: *Neural computation* 25.3 (2013), pp. 626–649.
- [131] Guillaume Hennequin, Tim P Vogels, and Wulfram Gerstner. “Optimal control of transient dynamics in balanced networks supports generation of complex movements”. In: *Neuron* 82.6 (2014), pp. 1394–1406.
- [132] Ryan C Williamson et al. “Bridging large-scale neuronal recordings and large-scale network models using dimensionality reduction”. In: *Current opinion in neurobiology* 55 (2019), pp. 40–47.
- [133] David Sussillo et al. “A neural network that finds a naturalistic solution for the production of muscle activity”. In: *Nature neuroscience* 18.7 (2015), pp. 1025–1033.
- [134] Maziar Raissi and George Em Karniadakis. “Hidden physics models: Machine learning of nonlinear partial differential equations”. In: *Journal of Computational Physics* 357 (2018), pp. 125–141.
- [135] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. “Multistep neural networks for data-driven discovery of nonlinear dynamical systems”. In: *arXiv preprint arXiv:1801.01236* (2018).
- [136] Ricky TQ Chen et al. “Neural ordinary differential equations”. In: *Advances in neural information processing systems* 31 (2018).
- [137] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. “Augmented neural odes”. In: *Advances in Neural Information Processing Systems* 32 (2019).
- [138] Patrick Kidger et al. “Neural controlled differential equations for irregular time series”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6696–6707.
- [139] James Morrill et al. “Neural rough differential equations for long time series”. In: *International Conference on Machine Learning*. PMLR, 2021, pp. 7829–7838.
- [140] Mathieu Chalvidal et al. “Go with the flow: Adaptive control for neural odes”. In: *arXiv preprint arXiv:2006.09545* (2020).
- [141] Lu Lu et al. “Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators”. In: *Nature Machine Intelligence* 3.3 (2021), pp. 218–229.

Chapter 2

Relevant Mathematical Background

Providing an in-depth mathematical introduction to the emerging field of deep dynamical systems is beyond the scope of this thesis. We instead refer the interested reader to [1]. In this section, we outline some basic mathematical concepts and ideas that would serve the reader well in the following chapters.

2.1 Some basic definitions

Definition 2.1.1 (Map). A map is defined as a deterministic evolution rule with discrete time on continuous state space. The evolution is simply given by iterating through this map. In the following definition, X is any topological space.

$$f : X \rightarrow X$$

For each initial condition $x_0 \in X$, the map f will give us the new state of the system after one application of the process.

$$x_1 = f(x_0)$$

Since $x_1 \in X$ (by definition of f), we can apply the map again to x_1 to get x_2 , which gives us the system's state after *two* applications of the process. More generally, the map f can be iterated as follows:

$$f^n := \underbrace{f \circ f \circ \dots \circ f}_{n \text{ times}}$$

, where $\circ$ is the function composition operator. Thus the n^{th} image of x_0 is given by

$$x_n = f^n(x_0)$$

In this thesis, we will use the notion of maps extensively in the context of deep recurrent neural networks (RNNs). Any RNN can be simply thought of as a highly non-linear map parameterized by a set of weights.

Definition 2.1.2 (Orbits). The forward orbit of $x_0 \in X$ is the set

$$\mathcal{O}^+(x_0) := \{x_n\}_{n \in \mathbb{N}}$$

In other words, the orbit of an initial value is simply the path that is specified under successive iterations of the map f in forward time. We also interchangeably refer to the orbit as a “trajectory”. The associated notion of a *reverse* orbit also exists in backward time if the inverse map f^{-1} is well defined.

Definition 2.1.3 (Fixed points). A point $p \in X$ is called a fixed point if and only if

$$f(p) = p$$

A more general definition is that of a *periodic* point. $p \in X$ is a periodic point with a period of $k \in \mathbb{N}$ if

$$p = f^k(p)$$

Trivially, fixed points are also periodic points with $k = 1$. A major focus of this review will be in identifying and characterizing fixed points of complex non-linear systems. While fixed points are useful, especially since dynamics around them can be “linearized”, periodic points provide insights into other forms of dynamics, including limit cycles.

Definition 2.1.4 (Flow). A flow is essentially a map that is continuously differentiable with respect to time.

$$\phi : \mathbb{R} \times X \rightarrow X$$

Extending the notation we have developed for orbits, we can write the orbit of a flow ϕ with an initial state of $x(0) \in X$ as

$$x(t) = \phi(t, x(0))$$

By definition, flows obey the following properties:

1. Identity: $\phi(0, x) = x$
2. Transitivity: $\phi(t + s, x) = \phi(t, \phi(s, x))$
3. Differentiability: $\left. \frac{d}{dt} \phi(t, x) \right|_{t=0} = Y(x)$

The differentiability criterion specifies that the derivative operator of a flow defines a vector field $Y(x)$ which in turn can be construed as solutions of ordinary differential equations (ODEs). The marriage between ODEs and deep neural networks has recently created a new research paradigm: Neural ODEs [2–4] and their offshoots [1, 5, 6]. Reviewing this literature is beyond the scope of this thesis. Instead, here we will focus on understanding techniques to identify and characterize temporal properties of dynamical systems. We start with the continuous-time formulation of differential equations and review the criterion for determining a given system’s “stability,” and then move on to the discrete-time descriptions of dynamical systems.

2.2 Linear stability of ODEs: The 1-Dimensional scenario

Consider a one dimensional ($x \in \mathbb{R}$) autonomous (t is not an explicit parameter of f) differential equation in the Euclidean plane given by

$$\frac{dx}{dt} = f(x)$$

Let $p \in \mathbb{R}$ be a fixed point of the system. The fixed point p is considered *stable* if given $\epsilon > 0$, we can find a $\delta > 0$ such that $\|x(t) - p\| < \epsilon$ whenever $\|x(t_0) - p\| < \delta$ for all $t \geq t_0$. Consider a small perturbation $\xi(t)$ away from p .

$$x(t) = p + \xi(t)$$

By taking the derivative with respect to time on both sides of this equation, we get

$$\frac{dx(t)}{dt} = \frac{d\xi(t)}{dt}$$

By definition of the differential system, we know that the left-hand side of this equation is simply $f(x)$.

$$\frac{d\xi(t)}{dt} = f(x) = f(p + \xi)$$

Taking the Taylor expansion of f around p we get,

$$\frac{d\xi(t)}{dt} = f(p) + \xi(t)f'(p) + \frac{\xi(t)^2}{2}f''(p) + \dots$$

Ignoring the second- and higher-order terms and taking into account that $f(p) = 0$ we get

$$\frac{d\xi(t)}{dt} = \xi(t)f'(p)$$

This is simply a first-order differential equation that we can solve using the linear separability technique as follows

$$\int \frac{1}{\xi(t)} d\xi(t) = \int f'(p) dt$$

$$\xi(t) = ae^{f'(p)t}$$

where a is a constant. From this, we can observe that as $t \rightarrow \infty$, $\xi(t)$ (the perturbation) will increase exponentially (unstable trajectories) if $f'(p) > 0$ and decrease exponentially (stable trajectories) if $f'(p) < 0$.

2.3 Linear stability of vector-valued ODEs

We can now apply the same treatment to vector-valued flows, albeit with minor modifications as necessitated by multivariate calculus. Consider a set of autonomous differential equations on $\mathbf{x} \in \mathbb{R}^n$ by $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ as

$$\frac{d\mathbf{x}(t)}{dt} = \begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \\ \vdots \\ \frac{dx_n}{dt} \end{bmatrix} = f(\mathbf{x})$$

Suppose that $\mathbf{x}^*$ is a fixed point of the system, i.e., $f(\mathbf{x}^*) = 0$. We can (similar to the 1-D analysis) take the multivariate Taylor expansion of f around $\mathbf{x}^*$ to get

$$\begin{aligned}\frac{d\mathbf{x}(t)}{dt} &= f(\mathbf{x}^*) + \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*) + \dots \\ \frac{d\mathbf{x}(t)}{dt} &\approx \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*)\end{aligned}$$

The partial derivative here of f with respect to $\mathbf{x}$ is referred to as the *Jacobian* of f . We reiterate the difference between a classic time derivative (such as the one on the left-hand side of the ODE) versus the Jacobian, which defines local rules of the function and not of the state variable. The common notation is to write the Jacobian of a function f evaluated at the point $\mathbf{x}^*$ as $J_f(\mathbf{x}^*)$. Mathematically speaking, the Jacobian is a matrix of partial derivatives.

$$J_f(\mathbf{x}^*) = \begin{bmatrix} \left. \frac{\partial f_1}{\partial x_1} \right|_{x_1^*} & \left. \frac{\partial f_1}{\partial x_2} \right|_{x_2^*} & \cdots & \left. \frac{\partial f_1}{\partial x_n} \right|_{x_n^*} \\ \left. \frac{\partial f_2}{\partial x_1} \right|_{x_1^*} & \left. \frac{\partial f_2}{\partial x_2} \right|_{x_2^*} & \cdots & \left. \frac{\partial f_2}{\partial x_n} \right|_{x_n^*} \\ \vdots & \vdots & \ddots & \vdots \\ \left. \frac{\partial f_n}{\partial x_1} \right|_{x_1^*} & \left. \frac{\partial f_n}{\partial x_2} \right|_{x_2^*} & \cdots & \left. \frac{\partial f_n}{\partial x_n} \right|_{x_n^*} \end{bmatrix}$$

Let us define a small perturbation from $\mathbf{x}^*$ as $\delta\mathbf{x} = \mathbf{x} - \mathbf{x}^*$. By observing that $\frac{d\delta\mathbf{x}(t)}{dt}$ is simply $\frac{d\mathbf{x}(t)}{dt}$ we can arrive at the following

$$\frac{d\delta\mathbf{x}(t)}{dt} = J_f(\mathbf{x}^*)\delta\mathbf{x}$$

As in the 1-Dimensional case, we can observe that to quantify the dynamics of the perturbations around this fixed point, it suffices to solve the above ODE (since the Jacobian is a constant matrix). By some rudimentary algebra, we can arrive at the solution

$$\delta\mathbf{x}(t) = e^{J_f(\mathbf{x}^*)t} \mathbf{c}$$

where $\mathbf{c} \in \mathbb{R}^n$ is simply a constant vector that can be identified by initial conditions to the ODE. Now, how can we determine the dynamics of the perturbation from this equation? Remember that the Jacobian is a matrix (square in our example), and the exponential series for a generic matrix $A \in \mathbb{R}^{n \times n}$ is given by

$$e^{At} = I + At + \frac{1}{2!}(At)^2 + \frac{1}{3!}(At)^3 + \dots$$

Let us consider the well-behaved scenario of the matrix A having n linearly independent eigenvectors. A can then be diagonalized as $A = V\Lambda V^{-1}$, where the columns of V are the eigenvectors of A and Λ is a diagonal matrix with the eigenvalues of A in the primary diagonal. Substituting this into the exponential series, we get

$$\begin{aligned}e^{At} &= I + V\Lambda V^{-1}t + \frac{1}{2!}(V\Lambda^2 V^{-1})t^2 + \frac{1}{3!}(V\Lambda^3 V^{-1})t^3 + \dots \\ e^{At} &= V(I + \Lambda t + \frac{1}{2!}\Lambda^2 t^2 + \frac{1}{3!}\Lambda^3 t^3 + \dots)V^{-1}\end{aligned}$$

$$e^{At} = Ve^{\Lambda t}V^{-1}$$

Given that Λ is a diagonal matrix, we can simplify this even further as

$$e^{At} = V \begin{bmatrix} e^{\lambda_1 t} & 0 & \dots & 0 \\ 0 & e^{\lambda_2 t} & \dots & 0 \\ & & \ddots & \\ 0 & 0 & \dots & e^{\lambda_n t} \end{bmatrix} V^{-1}$$

where $\lambda_1, \lambda_2, \dots, \lambda_n$ are the eigenvalues of A . Applying this logic to the solution to the ODE for $\delta \mathbf{x}(t)$ we get

$$\delta \mathbf{x}(t) = c_1 e^{\lambda_1 t} \mathbf{v}_1 + c_2 e^{\lambda_2 t} \mathbf{v}_2 + \dots + c_n e^{\lambda_n t} \mathbf{v}_n$$

Observing the relationship between the growth of the perturbation $\delta \mathbf{x}(t)$ and the eigenvalues of the Jacobian matrix brings us to the following important theorem.

Theorem 1. The equilibrium point $\mathbf{x}^*$ of the differential equation given by $\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x})$ is stable only if all the eigenvalues of $J_f(\mathbf{x}^*)$ are negative or have negative real parts. If at least one of the eigenvalues has a positive real part, then the equilibrium point is unstable.

2.4 Dynamics of 1-Dimensional maps

First, let us consider 1-D *linear* maps, which are the simplest class of dynamical systems. In fact, the only form of 1-D linear maps are the scalar maps $F : \mathbb{R} \rightarrow \mathbb{R}$, given by

$$F(x) = ax,$$

for some $a \in \mathbb{R}$. Iterates of this map are simply given by

$$F^n(x) = a^n(x),$$

for $n \in \mathbb{N}$. The dynamics are clearly governed by a in this case. It is easy to see that for $a \in (0, 1)$, the map eventually converges to 0 and hence is stable. On the other hand, if $|a| > 1$, the map is unstable. There are a few special cases one must take note of. In the case of $a = 0$, the map is non-invertible. That is, given the n -th iterate of the system, we wouldn't be able to identify the starting point. In the case of $a = \pm 1$, the map is non-hyperbolic.

One-dimensional *non-linear* maps offer a greatly expanded scope. In this case, we can write the iteration rule as $x_{t+1} = f(x_t)$, where $x_n \in \mathbb{R}$ and f is known as the generator function. Analogous to the continuous formulation, the stability of the 1-D map around a given fixed point is estimated by the slope of the generator function, $f'(x)$, evaluated at that point. If $|f'(x^*)| < 1$, then x^* is an attracting fixed point and if $|f'(x^*)| > 1$, x^* is a repelling fixed point. Importantly, if $|f'(x^*)| = 1$, we cannot make any guarantees about stability. Some illustrative examples are shown in Fig. 2.1.

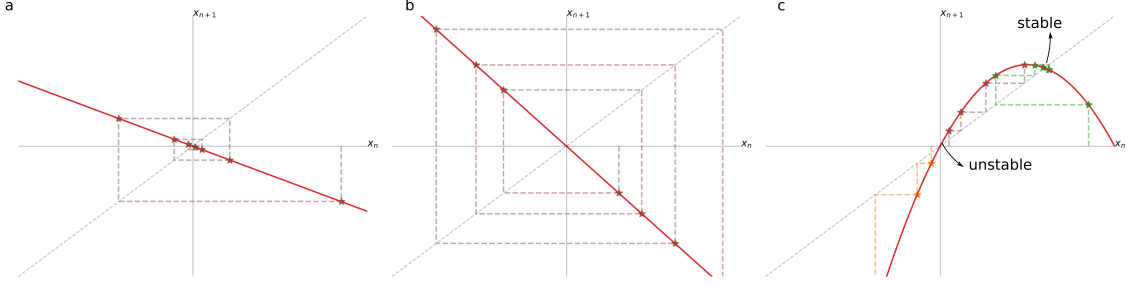


Figure 2.1: **Visualizing the dynamics of 1-D maps using cobweb plots.** Cobweb plots are a simple visualization strategy to trace the evolution of the state as we iterate through a map. Here these trajectories are shown as dashed lines. The gray dashed line is the line of unity. (a) The linear system $x_{n+1} = -0.5x_n$ has a fixed point at $x = 0$ that is stable. (b) The linear system $x_{n+1} = -1.2x_n$ also has a fixed point at $x = 0$ but is unstable. (c) The logistic map is a 1-D non-linear system. Here we show the dynamics for the map $x_{n+1} = 2.5x_n(1 - x_n)$. The fixed point at $x = 0$ is unstable, while the one at $x = 1$ is stable.

2.5 Asymptotic behavior of linear dynamical systems

Several applications in neuroscience involve characterizing the long-time-horizon behavior of linear dynamical systems. Let us consider the linear map

$$x_{t+1} = Ax_t,$$

where the coefficient matrix $A \in \mathbb{R}^{n \times n}$ and the state variable $x_t \in \mathbb{R}^n$. The simplicity of linear systems allows us to obtain the following closed-form solution

$$x_t = A^t x_0,$$

where x_0 is the initial state of the system. We are interested in studying what happens when $t \rightarrow \infty$. Under the assumption that the matrix A has a full set of independent eigenvectors $v_1, v_2, \dots, v_n$, we can rewrite x_0 with these eigenvectors as a basis set.

$$x_t = A^t(c_1 v_1 + c_2 v_2 + \dots + c_n v_n)$$

From the definition of eigenvectors, we also know that

$$A^t c_1 v_1 = A^{t-1} c_1 (A v_1) = A^{t-1} c_1 \lambda_1 v_1 = A^{t-2} c_1 \lambda_1 (A v_1) = A^{t-2} c_1 \lambda_1^2 v_1 = \dots = c_1 \lambda_1^t v_1$$

where λ_1 is the eigenvalue associated with the eigenvector v_1 . Distributing this property, we get

$$x_t = c_1 \lambda_1^t v_1 + c_2 \lambda_2^t v_2 + \dots + c_n \lambda_n^t v_n$$

Without loss of generality, we assume that $|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|$.

$$x_t = \lambda_1^t \left(c_1 v_1 + c_2 \left(\frac{\lambda_2}{\lambda_1} \right)^t v_2 + \dots + c_n \left(\frac{\lambda_n}{\lambda_1} \right)^t v_n \right)$$

$$\lim_{t \rightarrow \infty} x_t = \lim_{t \rightarrow \infty} \lambda_1^t c_1 v_1$$

The dominant eigenvalue λ_1 thus controls the asymptotic behavior of the system. However, we can also analyze the behavior of the system component-wise, i.e., the evolution along the direction of each eigenvector. In general, if $|\lambda| < 1$ the dynamics shrink along the corresponding direction. Similarly, if $|\lambda| > 1$, the dynamics expand with time. If $|\lambda| = 1$, then that component is conserved. In some cases, the dynamical system can have complex conjugate eigenvalues. We reparameterize them in polar coordinates as $\lambda = re^{i\theta}$ to analyze long-term behavior. Complex eigenvalues denote that the system's state is rotating around the origin in its phase space. If $|r| < 1$, the rotations have a shrinking amplitude resulting in a converging spiral. Similarly, when $|r| > 1$, the rotations have an expanding amplitude signifying a diverging spiral. If $|r| = 1$, the system's dynamics are stably rotating with a constant amplitude.

2.6 Linear stability of nonlinear dynamical systems

Analyzing the asymptotic properties of nonlinear systems is a very complex endeavor since such systems can express chaotic dynamics. However, we can study the “local” dynamics of these systems around equilibrium points. The mathematical background material we have covered until this point sets us up for this purpose. To recap, the reason we care about linear stability analysis is that deep recurrent neural networks are essentially discrete-time nonlinear dynamical systems whose dynamics we wish to control.

Consider the following nonlinear map

$$x_{t+1} = F(x_t)$$

In its most generic case, F can be any arbitrarily complex function. If this system has an equilibrium point x^* , then from Definition 2.1.3, we have

$$x^* = F(x^*)$$

Analogous to the ODE stability analysis, we proceed by adding an infinitesimal perturbation to this equilibrium point and quantify the behavior of the system. Say we choose $x_t = x^* + \Delta x_t$ and perform a Taylor expansion of F around x^* , we get

$$F(x^* + \Delta x_t) = F(x^*) + \left. \frac{\partial F}{\partial x} \right|_{x^*} \Delta x_t + \frac{1}{2!} \left. \frac{\partial^2 F}{\partial x^2} \right|_{x^*} (\Delta x_t)^2 + \dots$$

$$F(x^* + \Delta x_t) \approx F(x^*) + \left. \frac{\partial F}{\partial x} \right|_{x^*} \Delta x_t$$

$$x^* + \Delta x_{t+1} \approx x^* + \left. \frac{\partial F}{\partial x} \right|_{x^*} \Delta x_t$$

$$\Delta x_{t+1} \approx \left. \frac{\partial F}{\partial x} \right|_{x^*} \Delta x_t$$

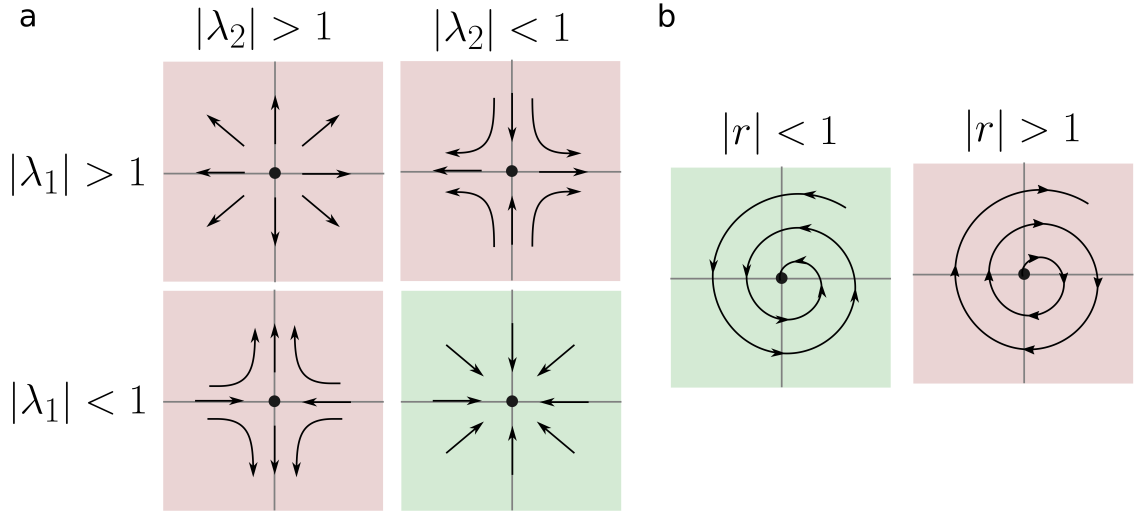


Figure 2.2: **Linear stability of equilibria in a 2-D dynamical system** The local dynamics of a system around a fixed point is fully specified by the eigenvalues of its Jacobian matrix. (a) In the case of real eigenvalues, we have four common scenarios. When both the eigenvalues are of magnitude greater than 1, we have unstable local dynamics, and when both have a magnitude smaller than 1, we have stable dynamics. In the other two scenarios, we have what's called a saddle point. This generally means that the dynamics attract along one axis and repel along the other. In the special case of either one or both eigenvalues equal to 1, we need to explicitly check for Lyapunov stability. (b) In the case of complex conjugate eigenvalues, we have rotational dynamics around the fixed point. Whether it's an attracting or repelling spiral depends on the magnitude of the complex vector. If the magnitude of the complex eigenvalue is equal to 1, then we have stable, fixed amplitude rotations.

This result informs us (as in the ODE case) that the growth of the perturbation is determined by the Jacobian matrix $\frac{\partial F}{\partial x}|_{x^*}$. As we did before, a spectral decomposition here would reveal the importance of the eigenvalues of this matrix in determining the stability of this system. For the reader's reference, common patterns of local dynamics around a fixed point in a 2-D dynamical system are shown in Fig. 2.2.

Naturally, in higher dimensions, one can extend these patterns to various other forms. Notably significant to computational neuroscience are networks of N neurons whose global dynamics stabilize in a D -dimensional space where usually $D \ll N$ [7]. Attractor networks can have a globally unique fixed point (also known as a point attractor) or have entire regions (subspaces) in their state space that serve as basins of attraction or repulsion. This subspace can be a line, ring, plane, or even a complex data manifold. Moreover, in high-dimensional systems, it is also common to observe chaotic dynamics that are highly expressive but hard to quantify or control.

2.7 Contraction maps

Contractors are mathematical operators that facilitate converging dynamics. Here, we will review standard results that link contraction maps to the existence of fixed points.

Definition 2.7.1. Let (X, d) be a metric space. A sequence $x_1, x_2, \dots$ is *Cauchy* if for every $\epsilon > 0$, $\exists N$ such that for $m, n > N$ we have $d(x_m, x_n) < \epsilon$. The space X is *complete* if every Cauchy sequence in X converges to an element that is present in X .

Definition 2.7.2. Let (X, d) be a metric space. $f : X \rightarrow X$ is a contraction if there exists $\lambda \in (0, 1)$ such that $\forall x, y \in X$,

$$d(f(x), f(y)) \leq \lambda d(x, y)$$

Theorem 2 (Contraction mapping theorem). If X is a complete metric space and $f : X \rightarrow X$ is a contraction, then there exists a unique, globally attracting fixed point $p \in X$.

Proof. By the definition of a complete metric space, we have that every Cauchy sequence converges. That is, for any initial x_0 , there exists a point p such that $x_n \rightarrow p$. Let us define $y_n := f(x_n)$. But we know that $f(x_n) = x_{n+1}$. As $n \rightarrow \infty$, $x_{n+1} \rightarrow p$ and hence $y_n \rightarrow p$. Assuming the continuity of f , we have

$$\begin{aligned} f\left(\lim_{n \rightarrow \infty} x_n\right) &= f(p) \\ f(p) &= f\left(\lim_{n \rightarrow \infty} x_n\right) = \lim_{n \rightarrow \infty} f(x_n) = \lim_{n \rightarrow \infty} y_n = p \end{aligned}$$

Thus we have $f(p) = p$ which is the definition of a fixed point.

Let us assume that we have two distinct fixed points p and $\hat{p}$. Then we would have,

$$d(f(p), f(\hat{p})) = d(p, \hat{p})$$

which is a violation of the contraction property of f . This is a contradiction and hence proves the uniqueness of the fixed point p .

2.8 References

- [1] Guan-Horng Liu and Evangelos A Theodorou. “Deep learning theory review: An optimal control and dynamical systems perspective”. In: *arXiv preprint arXiv:1908.10920* (2019).
- [2] Ricky TQ Chen et al. “Neural ordinary differential equations”. In: *Advances in neural information processing systems* 31 (2018).
- [3] Stefano Massaroli et al. “Dissecting neural odes”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 3952–3963.
- [4] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. “Augmented neural odes”. In: *Advances in Neural Information Processing Systems* 32 (2019).
- [5] Mathieu Chalvidal et al. “Go with the flow: Adaptive control for neural odes”. In: *arXiv preprint arXiv:2006.09545* (2020).
- [6] Patrick Kidger et al. “Neural controlled differential equations for irregular time series”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6696–6707.
- [7] Daniel J Amit and Daniel J Amit. *Modeling brain function: The world of attractor neural networks*. Cambridge university press, 1989.

Chapter 3

The Balance between Flexibility and Stability

3.1 Synopsis

Primate vision depends on recurrent processing for reliable perception [1–3]. A growing body of literature also suggests that recurrent connections improve the learning efficiency and generalization of vision models on classic computer vision challenges. Why then, are current large-scale challenges dominated by feedforward networks? We posit that the effectiveness of recurrent vision models is bottlenecked by the standard algorithm used for training them, “back-propagation through time” (BPTT), which has $\mathcal{O}(N)$ memory-complexity for training an N step model. Thus, recurrent vision model design is bounded by memory constraints, forcing a choice between rivaling the enormous capacity of leading feedforward models *or* trying to compensate for this deficit through granular and complex dynamics. Here, we develop a new learning algorithm, “contractor recurrent back-propagation” (C-RBP), which alleviates these issues by achieving constant $\mathcal{O}(1)$ memory-complexity with steps of recurrent processing. C-RBP is one of the first efforts to link the attractor dynamics perspective from computational neuroscience with large-scale recurrent neural networks in vision. We demonstrate that recurrent vision models trained with C-RBP can detect long-range spatial dependencies in a synthetic contour tracing task that BPTT-trained models cannot. We further show that recurrent vision models trained with C-RBP to solve the large-scale *Panoptic Segmentation* MS-COCO challenge outperform the leading feedforward approach, with fewer free parameters. C-RBP is a general-purpose learning algorithm for any application that can benefit from expansive recurrent dynamics. Code and data are available at <https://github.com/c-rbp>.

3.2 Introduction

Ullman (1984) famously theorized that humans reason about the visual world by composing sequences of elemental computations into “visual routines” [2]. It has been found that many of these visual routines, from perceptual grouping [4] to object categorization [5], depend on local and long-range recurrent circuits of the visual cortex [1, 3, 6]. Convolutional neural networks (CNNs) with recurrent connections – recurrent CNNs – also seem to learn visual routines that standard feedforward CNNs do not [7–14]. For example, consider the *Pathfinder* challenge in Fig. 3.1a, which asks observers to trace the contour extending from the white dot. Although *Pathfinder* is visually simple, clutter and variations in path shape make it difficult for feedforward CNNs to solve, even very deep residual networks [7, 8]. By contrast, a one-layer recurrent CNN can learn to solve *Pathfinder* by incrementally grouping paths from one end to the other, reminiscent of Gestalt-like visual routines used by human observers [7, 8, 15]. Others have found that the visual routines learned by recurrent CNNs on small computer vision datasets lead to better sample efficiency and out-of-distribution generalization than feedforward CNNs [6, 16, 17]. There is also evidence that primate visual decisions and neural responses elicited by natural images are best explained by recurrent CNNs [17–23]. Nevertheless, the great promise of recurrent CNNs has yet to translate into improvements on large-scale computer vision challenges like MS-COCO [24], which are dominated by feedforward CNNs.

A well-known limitation of recurrent CNNs is a memory bottleneck imposed by the standard learning algorithm, “back-propagation through time” (BPTT; [25]). The memory requirement of BPTT-trained models scales linearly with steps of processing since optimization involves propagating error through the full latent trajectory. This makes it difficult to develop recurrent CNNs that can rival the massive capacity of leading feedforward CNNs, which is critical for performance on challenges [26], while also simulating enough steps of processing to learn robust human-like visual routines.

Contributions. We develop a solution to the recurrent CNN memory bottleneck introduced by BPTT. Our work is inspired by recent successful efforts in memory-efficient approximations to BPTT for sequence modeling [27, 28]. Of particular interest is recurrent back-propagation (RBP), which exploits the stability of convergent dynamical systems to achieve constant memory complexity w.r.t. steps of processing [27, 29, 30]. This approach depends on models with stable dynamics that converge to a task-optimized steady state. However, we find that leading recurrent CNNs violate this assumption, and “forget” task information as they approach a steady state. While this pathology can be mitigated with hyperparameters that guarantee stability, these choices hurt model performance or “expressivity”. Thus, we make the observation that recurrent CNNs face a fundamental trade-off between stable dynamics and model expressivity that must be addressed before they can adopt efficient learning algorithms and compete on large-scale computer vision challenges.

- We derive a constraint for training recurrent CNNs to become both stable *and* expressive. We refer to this as the *Lipschitz-Constant Penalty* (LCP).

- We combine LCP with RBP to introduce “contractor-RBP” (C-RBP), a learning algorithm for recurrent CNNs with constant memory complexity w.r.t. steps of processing.
- Recurrent CNNs trained with C-RBP learn difficult versions of *Pathfinder* that BPTT-trained models cannot due to memory constraints, generalize better to out-of-distribution exemplars, and need a fraction of the parameters of BPTT-trained models to reach high performance.
- C-RBP alleviates the memory bottleneck faced by recurrent CNNs on large-scale computer vision challenges. Our C-RBP trained recurrent model outperforms the leading feedforward approach to the MS-COCO Panoptic Segmentation challenge with nearly 800K fewer parameters, and without exceeding the memory capacity of a standard NVIDIA Titan X GPU.

3.3 Background

We begin with a general formulation of the recurrent update step at $t \in \{1..N\}$ in an arbitrary layer of a recurrent CNN, which processes images with height H and width W .

$$h_{t+1} = F(x, h_t, w_F). \quad (3.1)$$

This describes the evolution of the hidden state $h \in \mathbb{R}^{H \times W \times C}$ through the update function F (a recurrent layer) with convolutional kernels $w_F \in \mathbb{R}^{S \times S \times C \times C}$, where C is the number of feature channels and S is the kernel size. Dynamics are also influenced by a constant drive $x \in \mathbb{R}^{H \times W \times C}$, which in typical settings is taken from a preceding convolutional layer. The final hidden state activity is either passed to the next layer in the model hierarchy, or fed into an output function to make a task prediction. The standard learning algorithm for optimizing parameters w_F w.r.t. a loss is BPTT, an extension of back-propagation to recurrent networks. BPTT is implemented by replicating the dynamical system in Eq. 3.1 and accumulating its gradients over N steps (Eq. 3.7). BPTT computes gradients by storing each h_t in memory during the forward pass, which leads to a memory footprint that increases linearly with steps.

Steady state-dependent learning rules There are alternatives to BPTT that derive better memory efficiency from strong constraints on model dynamics. One successful example is recurrent back-propagation (RBP), which optimizes parameters to achieve steady-state dynamics that are invariant to slight perturbations of input representations; a normative goal that echoes the classic Hopfield network [31]. When used with models that pass its test of stability, which we detail below, RBP memory complexity is *constant* and does not scale with steps of processing (the precision of dynamics). RBP is especially effective when a system’s steady states can be characterized by determining its Lyapunov function [27]. However, such analyses are generally difficult for non-convex optimizations, and not tractable for the complex loss landscapes of CNNs developed for large-scale computer vision challenges. RBP assumes that dynamics of the transition function $F(\cdot)$ eventually reach an equilibrium h^* as $t \rightarrow \infty$ (Eq. 3.2).

$$h^* = F(x, h^*, w_F) \quad (3.2) \quad \Psi(w_F, h) = h - F(x, h, w_F) \quad (3.3)$$

In other words, h^* is unchanged by additional processing. We can construct a function $\Psi(\cdot)$ such that the equilibrium h^* becomes its root (Eq. 3.3); i.e. $\Psi(w_F, h^*) = 0$. RBP leverages the observation that when differentiating Eq. 3.3, the gradient of steady state activity h^* w.r.t. the parameters of a stable dynamical system w_F can be directly computed with the Implicit Function Theorem [29, 30, 32, 33].

$$\frac{\partial h^*}{\partial w_F} = (I - J_{F, h^*})^{-1} \frac{\partial F(x, h^*, w_F)}{\partial w_F} \quad (3.4)$$

Here, J_{F, h^*} is the Jacobian matrix $\partial F(x, h^*, w_F)/\partial h^*$ (SI §3). In practice, the matrix $(I - J_{F, h^*})^{-1}$ is numerically approximated [27, 29]. RBP is designed for RNNs that pass its *constraint-qualifications* test for stability, where (i) $\Psi(\cdot)$ is continuously differentiable with respect to w_F and (ii) $(I - J_{F, h^*})$ is invertible. When these conditions are satisfied, as is often the case with neural networks for sequence modeling, RBP is efficient to compute and rivals the performance of BPTT [27].

3.3.1 Do recurrent CNNs pass the constraint-qualifications test of RBP?

We turn to the *Pathfinder* challenge [7, 8] to test whether RBP can optimize recurrent CNN architectures devised for computer vision (see 3.1a and Fig. S2 for examples). *Pathfinder* is an ideal test bed for recurrent vision models because they can more efficiently solve it than feedforward models.

- *Pathfinder* tests the ability of models to detect long-range spatial dependencies between features – identifying the target contour and tracing it from one end to the other. Feedforward architectures (like ResNets) need a sufficient amount of depth to learn such dependencies, leading to an explosion of parameters and learning problems [7, 8]. In contrast, recurrent CNNs can broaden receptive fields over steps of processing without additional processing layers (SI §3), but doing so requires maintaining task information across dynamics.
- *Pathfinder* is parameterized for fine-grained control over task difficulty. By lengthening or shortening target contours and clutter, we can generate more or less difficult datasets (Fig. S2) while controlling other perceptual variables that could introduce spurious image/label correlations.

In summary, recurrent CNNs that can solve *Pathfinder* need to learn a dynamic visual routine for detecting long-range spatial dependencies in images. BPTT-trained recurrent CNNs can do this [7, 8]. Here we test whether RBP-trained models can do the same.

Methods We trained the leading recurrent CNN for *Pathfinder*, the horizontal gated recurrent unit (hGRU; a complete model description is in SI §3.1), to solve a version where the target contour is 14-dashes long (Fig. 3.1a). We modified *Pathfinder* from its origins as a classification task [12] to a segmentation task, which makes it easier to interpret model dynamics and translate our findings

to the large-scale MS-COCO challenge examined in §3.4.2. The hGRU architecture consisted of (i) an input layer with 24 Gabor-initialized convolutional filters and one difference-of-Gaussians filter, followed by (ii) an hGRU with 15×15 horizontal kernels and 25 output channels, and finally (iii) a 1×1 convolutional “readout” that transformed the final hGRU hidden-state to a per-pixel prediction via batch normalization [34] and a 1×1 convolutional kernel. We began by testing four versions of the hGRU: one trained with BPTT for 6 steps, which was the most that could fit into the 12GB memory of the NVIDIA Titan X GPUs used for this experiment; and versions trained with RBP for 6, 20, and 30 steps. We also trained a fifth control model, a feedforward version of the 6 step BPTT-trained hGRU, where parameters were not shared across time (“feedforward CNN control”). The models were trained with Adam [35] and a learning rate of $3e-4$ to minimize average per-pixel cross entropy on batches of 32 images. Training lasted 20 epochs on a dataset of 200,000 images. Performance was measured after every epoch as the mean intersection over union (IoU) on a held-out test set of 10,000 images. We report the maximum IoU of each model on this test set.

Results The three hGRUs trained with RBP performed poorly on Pathfinder-14 (6 step: 0.50 IoU ; 20 step: 0.71 IoU ; 30 step: 0.70 IoU), far worse than a BPTT-trained hGRU (0.98 IoU). The RBP-hGRUs were also outperformed by the feedforward CNN control (0.78 IoU), although this control used 3 times more parameters than the hGRUs.

Why does an hGRU trained with RBP fail to learn *Pathfinder*? To address this question, we return to the constraint-qualifications test of RBP. The hGRU, like all models successfully optimized with gradient descent, satisfies condition (i) of the test: it is composed of a series of differentiable functions. This means that the problem likely arises from condition (ii), which requires the matrix $I - J_{F,h^*}$ to be invertible. Indeed, if $I - J_{F,h^*}$ is singular and not invertible, training dynamics will devolve into an unstable regime as we observed for RBP-hGRU training (Fig. S7). One way of guaranteeing an invertible $I - J_{F,h^*}$ is by forcing F to be a contractive map [33]. When this constraint is in place, we can invoke Banach fixed point theorem to ensure convergence to a unique fixed point starting from any initial hidden state h_0 . To elaborate, F implements a contraction map and will converge onto a unique fixed point if the following holds: $\forall h_i, h_j, x$, where $i, j \in \mathbb{R}_+$, $\exists \lambda \in [0, 1)$ such that

$$\|F(h_i; x, w_F) - F(h_j; x, w_F)\|_2 \leq \lambda \|h_i - h_j\|_2 \quad (3.5)$$

A straightforward way of forcing a contractive map is by building models with hyperparameters that are globally contractive (i.e., across the entire latent trajectory of dynamics), like squashing non-linearities (*sigmoid* and *tanh* [36, 37]). However, these same hyperparameters are suboptimal for computer vision, where unbounded non-linearities (like *ReLU*) are used because of their control over vanishing gradients in deep hierarchical networks and improving function expressivity [38, 39].

In principle, recurrent CNNs like convolutional LSTMs, that use *tanh* non-linearities, are better suited for RBP optimization than an hGRU, which uses soft unbounded rectifications (*softplus*). Indeed, we found that a 20-step convolutional LSTM (“convLSTM”; architecture detailed in SI §3.2) trained with RBP on Pathfinder-14 performs slightly better (0.73 IoU) than the RBP-trained

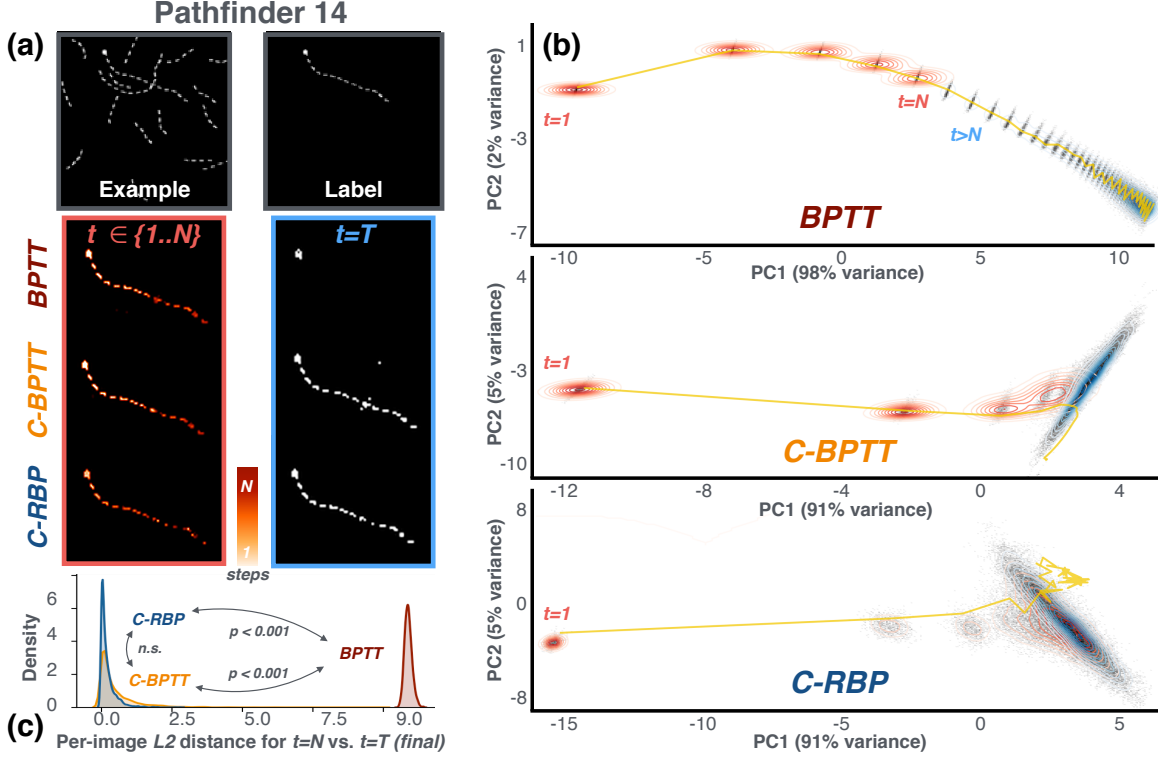


Figure 3.1: **Recurrent CNNs trained with backpropagation through time (BPTT) have unstable dynamics and forget task information.** This pathology is corrected by our *Lipschitz Coefficient Penalty* (LCP). (a) Incremental segmentations of horizontal gated unit (hGRU) models trained for N recurrent steps on Pathfinder-14. Heatmaps in the left column show the models tracing the target contour until $t=N=6$ steps. The right column depicts model predictions after $t=T=40$ steps. Segmentations from the BPTT-trained hGRU degenerate but models trained with LCP did not. Learning algorithms of LCP-trained models are “contractor-BPTT” (C-BPTT) and “contractor-RBP” (C-RBP). (b) Visualization of horizontal gated unit (hGRU) state spaces by projecting hidden states onto each model’s top-two eigenvectors. Grey dots are the 2D-histogram of projected hidden states, red contours are hidden state densities up to the task-optimized N steps, and blue contours are hidden state densities following that step. Dynamics for an exemplar image are plotted in yellow. BPTT-trained model dynamics diverge when $t > N$, but models trained with LCP did not. (c) Two-sample KS-tests indicate that the distance in state space between $t = N$ and $t = T$ hidden states is significantly greater for an hGRU trained with BPTT than an hGRU trained with C-BPTT or C-RBP (n.s. = not significant).

hGRU (0.71 *IoU*). At the same time, the RBP-trained convLSTM performs much worse than a 6-step BPTT-trained convLSTM (0.81 *IoU*) and a BPTT-trained hGRU (0.98 *IoU*). In other words, the convLSTM is more stable but less expressive than the hGRU. Furthermore, an hGRU with squashing non-linearities could not be trained reliably with RBP due to vanishing gradients. These findings raise the possibility that recurrent CNNs face a trade-off between “expressivity” needed for competing on large-scale computer vision challenges, and “stability” that is essential for using learning algorithms like RBP which rely on equilibrium dynamics.

Expressivity vs. stability To better understand the trade-off faced by recurrent CNNs, we examined the stability of an *expressive* recurrent CNN: the BPTT-trained hGRU, which outperformed all other architectures on Pathfinder-14. The hGRU solves Pathfinder by incrementally grouping the target contour, making it possible to visualize the evolution of its segmentation over time. By passing the model’s hidden states through its readout, we observed that task information vanishes from its hidden states after the task-optimized N -steps of processing (Fig. 3.1a; compare predictions at $t=N$ and $t=T$). These unstable dynamics were not specific to the BPTT-trained hGRU. We found similar results for a BPTT-trained convLSTM (Fig. S3), and when optimizing hGRUs with common alternatives to BPTT (Fig. S4). Next, we performed a state space analysis to measure model dynamics on all images in the Pathfinder-14 test set (method described in SI §3.3). The state space revealed a large divergence between hGRU hidden state activity at the task-optimized $t=6=N$ step vs. activity near steady state at $t=40=T$ (Fig. 3.1b). There was nearly as large of a difference between hidden states at $t=1$ and $t=N$ as there was between hidden states at $t=N$ and $t=T$.

3.4 Stabilizing expressive recurrent CNNs

While RNN stability is not necessary for all applications [40], it is critical for constant-memory alternatives to BPTT like RBP, which we hypothesized would improve recurrent CNN performance on large-scale vision challenges. Thus, we designed a “soft” architecture-agnostic constraint for learning local contraction maps, which balances model expressivity and stability over the course of training. Our goal was to derive a constraint to keep the largest singular value of $J_{F,h^*} < 1$, and force F to be locally contractive at h^* (SI §1.2; this contrasts with the global contraction across dynamics enforced by squashing non-linearities, which is problematic in computer vision as demonstrated in §3.3.1). However, the size of this Jacobian is quadratic in hidden state dimensions, making it too large to feasibly compute for recurrent CNNs. We overcome this limitation by introducing an approximation, our *Lipschitz Coefficient Penalty* (LCP), which constrains $(\mathbf{1} \cdot J_{F,h^*})_i < \lambda \forall i$, where i is a column index.

$$\|(\mathbf{1} \cdot J_{F,h^*} - \lambda)^+\|_2 \quad (3.6)$$

Here, $(\cdot)^+$ denotes element-wise rectification and $\lambda \in [0, 1)$ is the hand-selected Lipschitz constant which bounds $\|J_{F,h^*}\|_2$ and hence the degree of contraction in F . The LCP (Eq. 3.6) can be combined with any task loss for model optimization, and circumvents the need to explicitly determine the Jacobian J_{F,h^*} . We show in SI §1.2 that the vector-Jacobian product used in the LCP, which is efficiently computed through *autograd* in deep learning libraries, serves as an approximate upper bound on the spectral radius of the J_{F,h^*} . Note that because minimizing the LCP implicitly involves the Hessian of F , it is best suited for recurrent CNNs that use activation functions with defined second derivatives (like *softplus* for the hGRU).

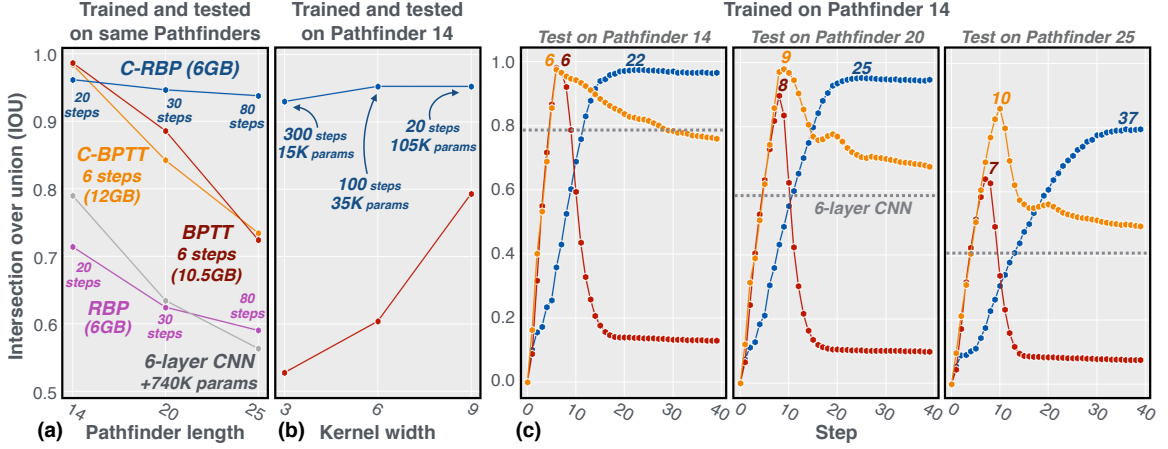


Figure 3.2: **Enforcing contraction in recurrent CNNs improves their performance and parameter efficiency and enables our constant-memory C-RBP learning algorithm.** (a) hGRU models were trained and tested on different versions of *Pathfinder*. Only the version trained with C-RBP maintained high performance across the three datasets. (b) C-RBP models can rely on recurrent processing rather than spatially broad kernels to solve long-range spatial dependencies. BPTT-trained models cannot practically do this due to their linear memory complexity. (c) LCP improves the stability of hGRU dynamics and, as a result, the generalization of learned visual routines for contour integration. Models were trained on Pathfinder-14, and tested on all three Pathfinder datasets. hGRUs trained with C-RBP and C-BPTT generalized far better than a version trained with BPTT or a 6-layer CNN control. Numbers above each curve denote the max-performing step.

3.4.1 Stable and expressive recurrent CNNs with constant memory complexity

Returning to *Pathfinder*, we tested whether the LCP stabilizes the dynamics of models trained with RBP or BPTT. LCP consists of a single hyperparameter, $\lambda \in [0, 1)$, which mediates the degree of contraction that is imposed. As $\lambda \rightarrow 0^+$, the local contraction at a fixed point becomes stronger leading to increased stability and reduced expressiveness. In general we find that λ is task dependent, but on *Pathfinder*, models performed well over many settings. We set $\lambda = 0.9$ on the experiments reported here, and adopt the methods described in §3.3.

Learning a task-optimal trade-off between stability and expressivity. A BPTT-trained hGRU constrained with LCP performed as well on Pathfinder-14 as one trained with BPTT with no constraint (0.98 *IoU* for both). However, a state space analysis of this “contractor-BPTT” (C-BPTT) hGRU revealed stable contractor dynamics, unlike the BPTT-trained hGRU (Fig. 3.1b). We took this success as evidence that hGRUs trained with LCP satisfy the RBP *constraint qualifications* test. We validated this hypothesis by training a 20-step hGRU with RBP and LCP, which we refer to as “contractor-RBP” (C-RBP). This model performs nearly as well as both BPTT and C-BPTT trained hGRUs (0.95 *IoU*), despite using approximately half the memory of either. The C-RBP hGRU also converged to a steady state that maintained task information (Fig. 3.1b; $t = T$), and like C-BPTT,

the distance between its task-optimized hidden state $t = N$ and steady state $t = T$ was smaller than the BPTT hGRU. Pairwise testing of these distances with 2-sample Kolmogorov–Smirnov (KS) tests revealed that the BPTT hGRU diverged from $t = N$ to $t = T$ significantly more ($\mu = 43.55$, $\sigma = 0.53$, $p < 0.001$) than either C-BPTT or C-RBP hGRUs ($\mu = 1.96$, $\sigma = 2.73$ and $\mu = 0.11$, $\sigma = 0.18$, respectively; Fig. 3.1c). We repeated these experiments for convLSTMs and found similar results, including C-RBP improving its performance on *Pathfinder* (Fig. S8). C-RBP therefore achieves our main goal of constant-memory training with performance on par with BPTT. However, in the following experiments, we demonstrate several qualities that make C-RBP preferable to BPTT.

C-RBP can solve *Pathfinder* tasks that BPTT cannot. We tested how recurrent CNNs performed on harder versions of *Pathfinder*, with 20- and 25-dash target paths, which forces them to learn longer-range spatial dependencies (Fig. S2). The linear memory complexity of BPTT caps the number of steps that can fit into memory, leading to poor performance on these datasets (Fig. 3.2a; BPTT and C-BPTT performance). Since C-RBP faces no such memory bottleneck, we were able to train 80-step C-RBP models to solve all versions of *Pathfinder*, achieving > 0.90 *IoU* on each dataset and vastly outperforming other models, including BPTT/RBP-trained hGRUs/convLSTMs, and the 6-layer CNN control model.

C-RBP models can achieve better parameter efficiency through extensive recurrent processing. *Pathfinder* is a test of contour integration, a visual routine for chaining arbitrarily small receptive fields together along the extent of the target contour. The instantaneous receptive field sizes of the recurrent CNNs used here are a function of convolutional kernel sizes. Thus, we hypothesized that models with small kernels could solve *Pathfinder* if they had sufficient steps of recurrent processing. Indeed, we found that hGRUs trained with C-RBP for 300 steps but only 3×3 kernels could solve Pathfinder-14 nearly as well as the baseline BPTT-trained hGRU with 15×15 kernels (Fig. 3.2b).

Stable dynamics improve the out-of-distribution generalization of visual routines. Visual routines such as contour integration derive long-range structure from repeated applications of a local grouping rule over steps of processing. This should lead to strong generalization on *Pathfinder*, even when testing out-of-distribution, or between datasets. For instance, increasing target contour length in *Pathfinder* affects the reaction time, but not accuracy, of human observers because they can solve the task through incremental grouping [8, 15]. If recurrent CNNs are learning similarly robust visual routines, a model trained on Pathfinder-14 should generalize to Pathfinder-20 and Pathfinder-25. To test this, we took hGRUs trained with BPTT, C-BPTT, or C-RBP on Pathfinder-14, and measured their performance on all versions of *Pathfinder* (Fig. 3.2c). Both C-BPTT and C-RBP trained hGRUs outperformed the BPTT-trained hGRU and feedforward control on each dataset, indicating that stable dynamics cause visual routines to generalize better. We found a similar improvement for C-RBP trained convLSTMs (Fig. S8). BPTT alternatives were not as effective as C-BPTT or C-RBP (Fig. S8).

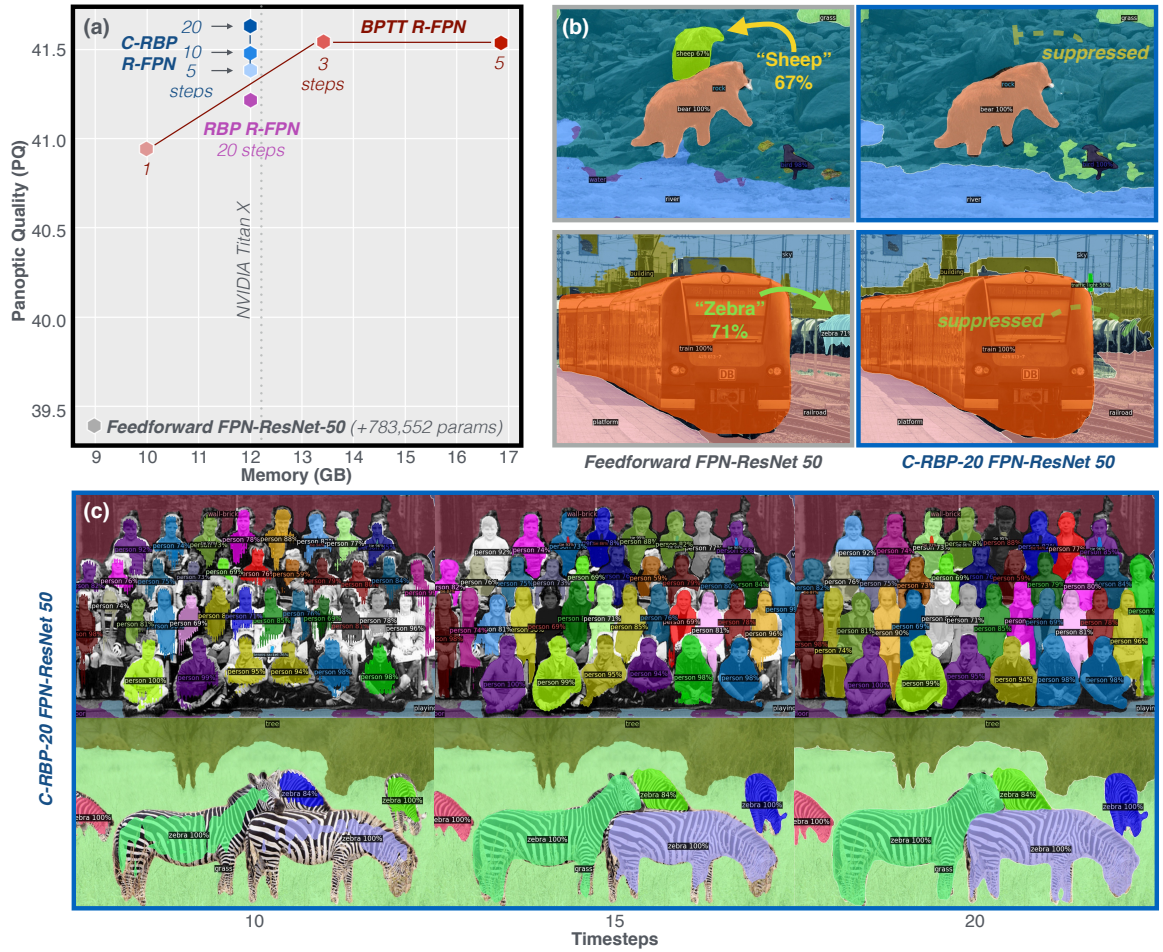


Figure 3.3: **C-RBP** trained recurrent vision models outperform the feedforward standard on MS-COCO Panoptic Segmentation despite using nearly 800K fewer parameters. (a) Performance of our recurrent FPN-ResNet 50 trained with **C-RBP** improves when trained with more steps of processing, despite remaining constant in its memory footprint. (b) Recurrent processing refines instance segmentations and controls false detections exhibited by the FPN-ResNet 50 (additional examples in SI). (c) Panoptic segmentation timecourses for an FPN-ResNet 50 trained with **C-RBP** for 20 steps.

3.4.2 Microsoft COCO Panoptic Segmentation

Our main motivation in this paper is to understand – and resolve – the memory bottleneck faced by recurrent CNNs for large-scale computer vision challenges. To date, there have not been competitive recurrent solutions to the MS-COCO Panoptic Segmentation Challenge, a difficult task which involves recognizing the semantic category and/or object instance occupying every pixel in an image [41]. However, we will show that the stable dynamics and constant memory efficiency of C-RBP allows us to construct recurrent models that outperform the feedforward standards.

Methods The leading approach to Panoptic Segmentation is an FPN Mask-RCNN [41] with a ResNet-50 or ResNet-101 layer backbone (referred to hereafter as FPN-ResNet-50 and FPN-ResNet-101). We developed recurrent extensions of these models, where we replaced the 4-layer Mask-RCNN head with an hGRU containing 3×3 kernels (SI §3.4). We call these models R-FPN-ResNet-50 and R-FPN-ResNet-101. MS-COCO Panoptic Segmentation involves two separate objectives: (i) instance segmentation of the discrete and enumerable “things” in an image, such as people and animals; and (ii) semantic segmentation of the remaining background elements in scenes, or “stuff” as it is known in the challenge. Models were trained to optimize both of these objectives with SGD+Momentum, a learning rate of 5e-2, and batches of 40 images across 24GB NVIDIA GTX GPUs (10 total). Model predictions on the COCO validation set were scored with Panoptic Quality (PQ), which is the product of metrics for semantic (IoU) and instance (F_1 score) segmentation [41]. Note that differences between models in PQ scores can be interpreted as a percent-improvement on instance and semantic segmentation for the total image. Recurrent models trained with LCP used $\lambda = 0.9$; training failed with higher values of this hyperparameter.

Results After replicating benchmark performance of an FPN-ResNet-50 (39.4PQ, 9GB memory; <https://github.com/crbp>), we evaluated our R-FPN models (Fig. 3.3a). We first trained versions of the R-FPN-ResNet-50 with BPTT, which used nearly 17GB of memory for 5-steps of processing. BPTT performance increased with recurrence until it plateaued at 3-steps (41.5PQ for both 3- and 5-step models). Next, we trained an R-FPN-ResNet-50 with RBP for 20 steps, and found that this model (41.22PQ, 12GB) performed better than the 1-step R-FPN-ResNet-50, but worse than a 2-step BPTT-trained model (41.45PQ). R-FPN-ResNet-50 models improved when they were trained with C-RBP. A 5-step model trained with C-RBP (41.39, 12GB) outperformed a 20-step model trained with RBP; a 10-step C-RBP model was similar (41.48PQ, 12GB) to the 3- and 5-step BPTT models; and a 20-step C-RBP model performed best (41.63PQ, 12GB). Importantly, each of these R-FPN-ResNet-50 models was more accurate than the feedforward FPN-ResNet-50 despite using 783,552 *fewer* parameters. We also found that a 20-step C-RBP R-FPN-ResNet-101 outperformed the feedforward FPN-ResNet-101 and BPTT-trained R-FPN-ResNet-101 models (Fig. S10).

There are two key qualitative improvements for the 20-step C-RBP R-FPN-ResNet-50 over the standard feedforward model. First, it suppressed false detections of the feedforward model, such as an over-reliance on texture for object recognition (Fig. 3.3b; other examples in Fig. S11-12). Second,

and most surprisingly, the C-RBP model learned to solve the task by “flood-filling” *despite no explicit constraints to do so* (Fig. 3.3c). There is extensive work in the brain sciences suggesting that human and primate visual systems rely on a similar algorithm for object segmentation, classically referred to as “coloring” [2, 42]. We validated the extent to which the C-RBP R-FPN-ResNet-50 discovered a “flood-filling” segmentation routine by tessellating the ground-truth object masks in images, then applying a standard flood-filling algorithm to the resulting map. Both the C-RBP R-FPN-ResNet-50 and this idealized flood-filling model exhibited similar segmentation strategies (Fig. 3.4).

3.4.3 Related work

Memory efficient learning algorithms BPTT’s memory bottleneck has inspired many efficient alternatives. A popular example is truncated back-propagation through time (TBPTT), which improves memory efficiency with a shorter time horizon for credit assignment. There are also heuristics for overcoming the BPTT memory bottleneck by swapping memory between GPU and CPU during training [43, 44], or “gradient checkpointing”, and recomputing a number of intermediate activities during the backwards pass [45–47]. Neural ODEs can train neural networks with continuous-time dynamics by optimizing through black box ODE solvers, but their computational demands and numerical issues result in poor performance on vision tasks ([48], see SI §4.1 for an expanded discussion). Deep equilibrium models (DEQ) are another constant-memory complexity learning algorithm with good performance on sequence modeling, rivaling the state of the art [28]. However, we found that training recurrent CNNs with DEQ is unstable, like with RBP (SI §4.1).

Stable RNNs Recurrent model “stability” is typically analyzed w.r.t. training, where exploding or vanishing gradients are a byproduct of gradient descent [40, 49]. Models are stable if their Jacobians are upper-bounded by λ (i.e., it is λ -contractive). One way to achieve this is with architecture-specific constraints on model parameters, which has worked well for generative adversarial networks (GANs) [50, 51] and sequence modeling LSTMs [40]. A simple alternative, is to directly penalize Jacobian approximations, as has been done for GANs [52–54]. However, these approximations have not been investigated for recurrent vision models or in the context of visual recognition.

3.5 Limitations and Future Directions

More work is needed to extend C-RBP to visual tasks other than the ones we analyzed here. For example, it is unclear whether C-RBP can extend to complex dynamical systems with limit cycles or strange attractors. It is possible that extensions are needed to transform C-RBP into a general method for balancing stability and expressivity when optimizing recurrent neural networks to explain arbitrary dynamical systems. We also did not examine complex recursive architectures where all recurrent layers are on the same dynamic “clock”, such as in the γ -net of [17]. We leave the details of this application to future work, which we anticipate will be especially impactful for computational neuroscience applications, where such hierarchical feedback models can explain biological data that

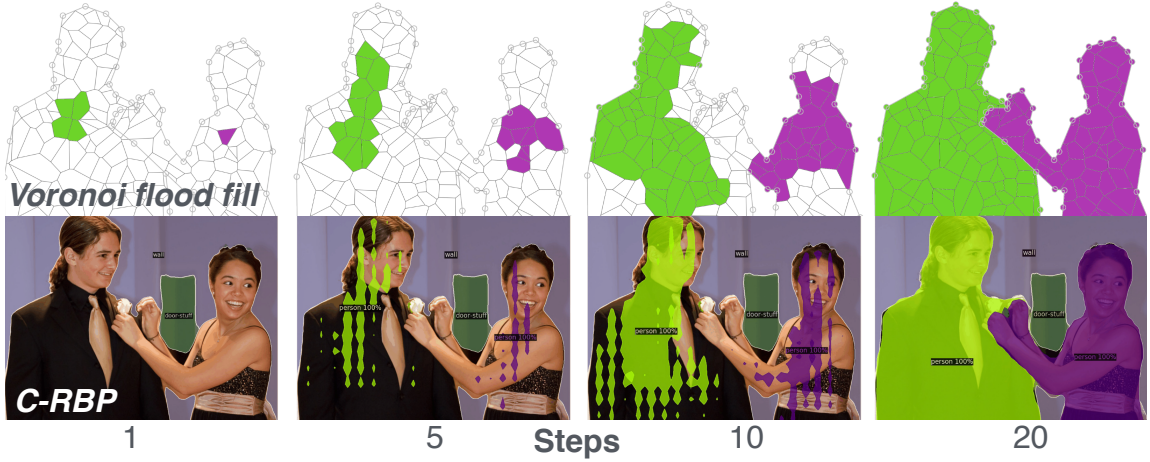


Figure 3.4: **C-RBP**-optimized models for Panoptic segmentation learn to implement a routine that resembles a flood-fill algorithm, despite no explicit supervision to do so. **Top** Flood-filling in a Voronoi-tessellation of object masks. **Bottom** The routine learned by a **C-RBP** R-FPN-ResNet-50.

feedforward models cannot [8, 13, 17, 19, 22]. Another open problem is extending our formulation to spatiotemporal tasks, like action recognition, tracking, and closed-loop control. These tasks are defined by a time-varying feedforward drive, which presents novel optimization challenges.

3.6 Conclusion

There is compelling evidence from biological and artificial vision that the visual routines which support robust perception depend on recurrent processing [1–3]. Until now, the enormous cost of training recurrent CNNs with BPTT has made it difficult to test the hypothesis that these models can learn routines that will improve performance on large-scale computer vision challenges, which are dominated by high-capacity feedforward CNNs. C-RBP can alleviate this bottleneck, enabling constant-memory recurrent dynamics, more parameter-efficient architectures, and systematic generalization to out-of-distribution datasets.

3.7 Extended background

3.7.1 Backpropagation through time (BPTT)

BPTT is the standard learning algorithm for optimizing recurrent parameters w_F w.r.t. a loss $\mathcal{L}(\tilde{y}, y)$. It is implemented by replicating a dynamical system and accumulating its gradients over N steps of processing (Eq. 3.7, $K = 0$). Given a recurrent function F parameterized by w_F , which maintains a latent state h_t for each time step t , BPTT is implemented by Eq. 3.7.

$$\frac{\partial \mathcal{L}}{\partial w_F} = \frac{\partial \mathcal{L}}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial h_T} \sum_{k=K}^{T-1} \left(\prod_{i=T-k}^{i=T-1} J_F(h_i) \right) \frac{\partial F}{\partial w_F}(x, h_{T-k}, w_F). \quad (3.7)$$

Here, $J_F(h_i)$ is the Jacobian of F at h on step i . Note that this algorithm stores every h_t in memory during the forward pass, causing a memory footprint that linearly increases with steps.

3.7.2 Lipschitz Coefficient Penalty

We designed the *Lipschitz Coefficient Penalty* (LCP) as a hyperparameter-agnostic regularization for forcing recurrent CNNs to learn contraction maps. As mentioned in the main text, LCP constrains the vector-Jacobian product $(\mathbf{1} \cdot J_{F,h^*})_i < 1 \forall i$, where i is a column index.

$$\|(\mathbf{1} \cdot J_{F,h^*} - \lambda)^+\|_2. \quad (3.8)$$

Here, $(\cdot)^+$ denotes element-wise rectification and $\lambda \in [0, 1)$ controls the degree of contraction in F . To derive LCP, we begin from the first-order Taylor expansion of $F(h)$,

$$F(h) \approx F(\bar{h}) + J_{F,\bar{h}} \cdot (h - \bar{h}) + \dots,$$

with which one can show:

$$\frac{\|F(h) - F(\bar{h})\|_2}{\|h - \bar{h}\|_2} \approx \frac{\|J_{F,\bar{h}} \cdot (h - \bar{h})\|_2}{\|h - \bar{h}\|_2}, \quad (3.9)$$

Recalling the necessary condition for F being a contractive map,

$$\|F(h) - F(\bar{h})\|_2 < \lambda \|h - \bar{h}\|_2, \quad (3.10)$$

we can observe that the right hand side of Eq. 3.9 must be less than or equal to $\lambda \in [0, 1)$ for any h sufficiently close to $\bar{h}$. Thus, $F(\cdot)$ will be λ -contractive *at least in the neighborhood* of $\bar{h}$ if the LHS is forced to be less than λ . We accomplish this goal by explicitly regularizing $\bar{h} = h^*$ over the course of training:

$$\|J_{F,h^*} \cdot \hat{v}\|_2 < 1, \quad (3.11)$$

for all unit vectors $\hat{v}$, which implies that the largest singular value of J_{F,h^*} must be less than 1. This is equivalent to requiring $\|J_{F,h^*}\|_2 < 1$. Note that $\|h - h^*\|_2$ may not necessarily be small for all h 's sampled along our trajectories, and so the Taylor approximation and hence Eq. (3.9) may not

hold. Nevertheless, in experiments on Pathfinder and Microsoft COCO our regularisation still yields reasonably stable convergence to fixed points.

Indeed, the matrix 2-norm is bounded from above and below by the 1-norm,

$$1/\sqrt{n} \|J_{F,h^*}\|_1 \leq \|J_{F,h^*}\|_2 \leq \sqrt{n} \|J_{F,h^*}\|_1, \quad (3.12)$$

where

$$\|J_{F,h^*}\|_1 = \max_i \sum_j |(J_{F,h^*})_{ij}|, \quad (3.13)$$

and n is the dimensionality of the Jacobian matrix. So if we can regularize and force $\sqrt{n} \|J_{F,h^*}\|_1$ to be below 1, then we can ensure that $F(\cdot)$ will be contractive. However, computing Jacobians of large matrices requires an enormous memory load, and it is far more efficient to compute vector-Jacobian products instead. We shall instead approximate the 1-norm by taking

$$\max_i \left| \sum_j (J_{F,h^*})_{ij} \right| = \max_i (\mathbf{1} \cdot J_{F,h^*})_i, \quad (3.14)$$

where $\mathbf{1}$ denotes the row vector with all entries being 1. We note that in using this approximation, the right inequality in (3.12) ceases to be a strict upper bound, but we find that the approximation works well in practice due to large n .

We regularise model training with this approximation by requiring that $(\mathbf{1} \cdot J_{F,h^*})_i < 1 \forall i$. This yields our *Lipschitz Coefficient Penalty* (LCP):

$$\|(\mathbf{1} \cdot J_{F,h^*} - \lambda)^+\|_2, \quad (3.15)$$

which can be added to any task loss. Here, $(\cdot)^+$ denotes element-wise rectification and $\lambda \in [0, 1)$ is a hand-selected constant controlling the bound on $\|J_{F,h^*}\|_2$ and hence the degree of contraction in F . See Fig. 3.5 for an example implementation of LCP in pytorch.

```

1 import torch
2
3 def LCP(last_state, second_last_state, mu=0.9):
4     """Implementing the Lipschitz Coefficient Penalty with autograd."""
5     norm_1_vect = torch.ones_like(last_state) # Compute vector-jacobian product
6     norm_1_vect.requires_grad = False
7     vj_prod = torch.autograd.grad(
8         last_state,
9         second_last_state,
10        grad_outputs=[norm_1_vect],
11        retain_graph=True,
12        create_graph=True,
13        allow_unused=True)[0]
14     vj_penalty = (vj_prod - mu).clamp(0) ** 2 # Clamp at mu and square
15     return vj_penalty.mean() # Minimize the average

```

Figure 3.5: Pytorch 1.4 code for computing the LCP.

3.8 Recurrent Back-prop

We review the Recurrent Back-Prop (RBP) learning algorithm of [29, 30]. Given a transition function F , which is parameterized by w_F and applied to the static drive x , hidden state h over $t \in \{1..N\}$ steps of processing: $h_{t+1} = F(x, h_t, w_F)$. We define a model readout, $\tilde{y} = G(h_T, w_G)$, where G is a task-optimized readout parameterized by w_G . We also introduce a loss $\mathcal{L}$ which yields a distance between predicted and ground-truth outputs. By differentiating the loss with respect to the weights, we obtain the gradients:

$$\frac{\partial \mathcal{L}_\infty}{\partial w_G} = \frac{\partial \mathcal{L}_\infty}{\partial y_\infty} \frac{\partial G(x; h^*, w_G)}{\partial w_G}, \quad (3.16)$$

$$\frac{\partial \mathcal{L}_\infty}{\partial w_F} = \frac{\partial \mathcal{L}_\infty}{\partial y_\infty} \frac{\partial y_\infty}{\partial h^*} \frac{\partial h^*}{\partial w_F}. \quad (3.17)$$

To obtain an expression for $\partial h^* / \partial w_F$, we first introduce the auxiliary function

$$\Psi(w_F, h) = h - F(x, h, w_F), \quad (3.18)$$

where, at a fixed point, $\Psi(w_F, h^*) = 0$. Differentiating with respect to w_F , we obtain

$$\begin{aligned} \frac{\partial \Psi(w_F, h^*)}{\partial w_F} &= \frac{\partial h^*}{\partial w_F} - \frac{dF(x, h^*, w_F)}{dw_F} \\ &= \left(I - \frac{\partial F(x, h^*, w_F)}{\partial h^*} \right) \frac{\partial h^*}{\partial w_F} - \frac{\partial F(x, h^*, w_F)}{\partial w_F} \\ &= 0. \end{aligned}$$

Rearranging yields

$$\frac{\partial h^*}{\partial w_F} = (I - J_{F, h^*})^{-1} \frac{\partial F(x, h^*, w_F)}{\partial w_F}, \quad (3.19)$$

where J_{F, h^*} is the Jacobian matrix $\partial F(x, h^*, w_F) / \partial h^*$. The implicit function theorem guarantees the existence and uniqueness of a function mapping w_F to h^* and hence of $\partial h^* / \partial w_F$ provided (i) $\Psi(w_F, h)$ is continuously differentiable and (ii) $(I - J_{F, h^*})$ is invertible. The main virtue of RBP is that the memory it requires to train the RNN is constant with respect to the granularity of dynamics (steps of processing).

3.9 Recurrent vision models

3.9.1 *Pathfinder*

Horizontal Gated Recurrent Units The hGRU is a recurrent CNN, which when placed on top of a conventional feedforward convolutional layer, implements long-range nonlinear interactions between the feedforward layer’s units [7]. These interactions take place over “horizontal connections” – a concept from neuroscience, in which anatomical connections between nearby cortical neurons (in retinotopic space) are the substrate for complex recurrent processing. Tracing back to its origins as a neural circuit model, the hGRU distinguishes itself from other recurrent CNNs as having two

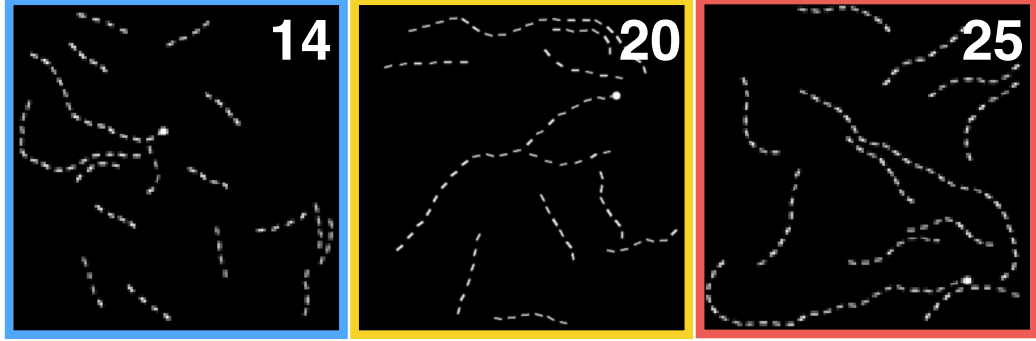


Figure 3.6: **Parameterically controlling task difficulty.** In our variation of the *Pathfinder* challenge [7, 8], we ask observers to segment the contour connected to the white dot. Recurrent CNNs can easily solve it by learning to incrementally “trace” the target from end to end. Target contours are made up of 14-, 20-, or 25-dashes.

distinct stages of processing with independent kernels in each. The first stage computes suppressive interactions, i.e., a unit at location (x, y) inhibits activity in a unit at location $(x + n, y)$, where n is a spatial offset between these units. The second stage computes facilitative interactions, i.e., a unit (x, y) excites activity in a unit at location $(x, y + n)$. The hGRU is governed by the following equations:

Stage 1:

$$\begin{aligned}
 \mathbf{A}^S &= U^S * \mathbf{H}[t-1] && \# \text{ Compute channel-wise selection} \\
 \mathbf{G}^S &= \text{sigmoid}(\mathbf{A}^S) && \# \text{ Compute suppression gate} \\
 \mathbf{C}^S &= BN(W^S * (\mathbf{H}[t-1] \odot \mathbf{G}^S)) && \# \text{ Compute suppression interactions} \\
 \mathbf{S} &= \left[\mathbf{Z} - \left[(\alpha \mathbf{H}[t-1] + \mu) \mathbf{C}^S \right]_+ \right]_+ , && \# \text{ Additive and multiplicative suppression of } \mathbf{Z}
 \end{aligned}$$

Stage 2:

$$\begin{aligned}
 \mathbf{G}^F &= \text{sigmoid}(U^F * \mathbf{S}) && \# \text{ Compute channel-wise recurrent updates} \\
 \mathbf{C}^F &= BN(W^F * \mathbf{S}) && \# \text{ Compute facilitation interactions} \\
 \tilde{\mathbf{H}} &= \left[\nu(\mathbf{C}^F + \mathbf{S}) + \omega(\mathbf{C}^F * \mathbf{S}) \right]_+ && \# \text{ Additive and multiplicative facilitation of } \mathbf{S} \\
 \mathbf{H}[t] &= (1 - \mathbf{G}^F) \odot \mathbf{H}[t-1] + \mathbf{G}^F \odot \tilde{\mathbf{H}} && \# \text{ Update recurrent state}
 \end{aligned}$$

where $BN(\mathbf{R}; \delta, \nu) = \nu + \delta \odot \frac{\mathbf{R} - \hat{\mathbb{E}}[\mathbf{R}]}{\sqrt{\widehat{\text{Var}}[\mathbf{R}] + \eta}}$.

Here, $\mathbf{H}, \mathbf{Z} \in \mathcal{R}^{X \times Y \times C}$ are the hidden state and static drive from a preceding convolutional layer, respectively, with height/width/channels X, Y, C . Suppressive interactions in Stage 1 are computed with $W^S \in \mathbb{R}^{E \times E \times C \times C}$, and facilitative interactions in Stage 2 are computed with $W^F \in \mathbb{R}^{E \times E \times C \times C}$, where E is the spatial extent of the horizontal connection kernel. In most of our experiments we set

$E = 15$, as in [7] (other kernel sizes were tested in our parameter efficiency analysis in Fig. 2b). The hGRU also contains gates to modulate input activity and interpolate the previous hidden state with the current step’s state, $U^S, U^F \in \mathbb{R}^{1 \times 1 \times C \times C}$. Steps of processing are indexed by $t \in \{1..N\}$, and rectification using *softplus* pointwise nonlinearities is denoted by $[\cdot]_+$, which ensures non-negativity in each stage, and hence, guarantees on suppression vs. facilitation. Lastly, we use batch normalization in the module to control exploding/vanishing gradients [55]. This introduces two learned kernels, $\delta, \nu \in \mathbb{R}^{1 \times 1 \times C}$, which control the scale and bias of normalization over input feature maps $\mathbf{R}$, and are shared across steps of processing (η is a small constant that protects divide-by-zero errors). As is standard in batch normalization, $\widehat{\mathbb{E}}$ and $\widehat{\text{Var}}$ are estimated on-line during training.

Convolution LSTM We use a standard implementation of convolutional LSTMs from [14]. These models used kernels with the same dimensions as those described above for the hGRU.

3.9.2 State space analysis

We analyzed the state space of recurrent models trained to solve *Pathfinder*. This classic technique from dynamical systems has shown promise for analyzing computations of task-optimized recurrent neural networks [56]. Our approach to visualizing model state spaces involved the following steps: **(i)** Extract model hidden states for steps $t \in \{1..T\}$ elicited by a Pathfinder-14 image. **(ii)** Reduce hidden state dimensionality with a global average pool across spatial dimensions, yielding C -dimensional vectors. **(iii)** Fit a PCA using the $t \in \{1..N\}$ task-optimized steps of processing. **(iv)** Project all $t \in \{1..T\}$ hidden states onto the extracted eigenvectors.

3.9.3 Panoptic Segmentation

As a proof-of-concept of C-RBP on large-scale computer vision challenges, we developed a straightforward recurrent extension to the leading feedforward approach to the MSCOCO Panoptic Segmentation challenge: the FPN-ResNet. This model uses a ResNet backbone (either 50- or 101-layers) pretrained on ImageNet, which passes its activities to a feature pyramid network (FPN; [57]). FPN activities are then sent to a linear readout for semantic segmentation (to identify the “stuff” in images) and a Mask-RCNN for instance segmentation (to identify and individuate the “things”). We replace the Mask-RCNN head, which consists of 4-layers of convolutions and linear rectifications, with a single hGRU module (for both our version and the standard, activities from this stage are next upsampled, rectified, and linearly transformed into predictions). The hGRU that we used is slightly different than the one for the *Pathfinder* challenge above. Batch normalization was replaced with group normalization [58], which is standard for Panoptic segmentation. We also used a modification of the input gate, following [17], which was found to improve performance for natural image processing. Standard feedforward FPN-ResNets were approximately twice as fast to train as our 20-step C-RBP R-FPN-ResNets. Models with ResNet-50 backbones took between one and two days to train, whereas models with ResNet-101 backbones took two and four days to train.

3.10 Extended discussion

3.10.1 Related work

Recurrent vision models There are many successful applications of recurrent CNNs in computer vision, including object recognition, segmentation, and super-resolution tasks [9–14, 59]. These models often augment popular feedforward CNN architectures with local (within a layer) and/or long-range (between-layer) recurrent connections.

Others have found that augmenting recurrent CNNs with connectivity patterns or objective functions that are inspired by the anatomy or the physiology of the visual cortex can improve performance in visual reasoning, prediction, and recognition of occluded objects [12–14, 59–63].

Lipschitz constraints for stable *training* There are many examples of using constraints on Lipschitz continuity to stabilize deep network training. This is especially popular for generative adversarial networks (GANs), where stability is enhanced by constraining the spectral norm of the weights [50, 51], or Jacobians of each layer of the discriminator [52], or through Monte Carlo estimation of the discriminator’s Jacobian [53, 54]. Penalizing the spectral norm of Jacobians can also yield better Auto Encoders [64], and adversarial robustness in CNNs [52]. In contrast to prior works on penalizing network Jacobians, in the current work we describe (i) an application to recurrent vision models, which (ii) enforces contractions only locally around equilibrium points (rather than globally across a hierarchy), which is a weaker constraint on model expressivity that still supports our key goal of stability during inference.

RNNs are notoriously challenging to train [49], and the classic solution is to constrain Lipschitz continuity by introducing learnable gates [40, 65]. It should be emphasized that our models also take advantage of gates, and while these control vanishing and exploding gradients to stabilize training, they are not sufficient to yield contractive mappings. More generally, it has been found that stability is critical to train RNNs that can solve sequence modeling tasks [66], but that stability is less critical for BPTT [40]. Other recent approaches induce stability via other architectural constraints, like weight orthogonalization via SVD [67].

Deep networks as ODEs Neural ODEs exploit the observation that the residual networks can be treated as a discrete-time ODE. By using black-box ODE solvers in the forward and backward passes of the network, this discretization can be taken towards zero [68]. These models are trained with back-propagation through a latent trajectory derived from an adjoint system, giving them constant memory efficiency w.r.t. the granularity of dynamics, and they have shown promise in modeling continuous-time data and normalizing flows. However, Neural ODEs face several issues for computer vision applications. (i) Neural ODEs are difficult to optimize because input-output mappings become arbitrarily complex over the course of training. (ii) The adjoint method is slow to compute. (iii) Neural ODEs require feature engineering to fit certain classes of non-linear data manifolds [48], and (iv) they (along with the recent Augmented Neural ODEs [48]) do not compare favorably to standard

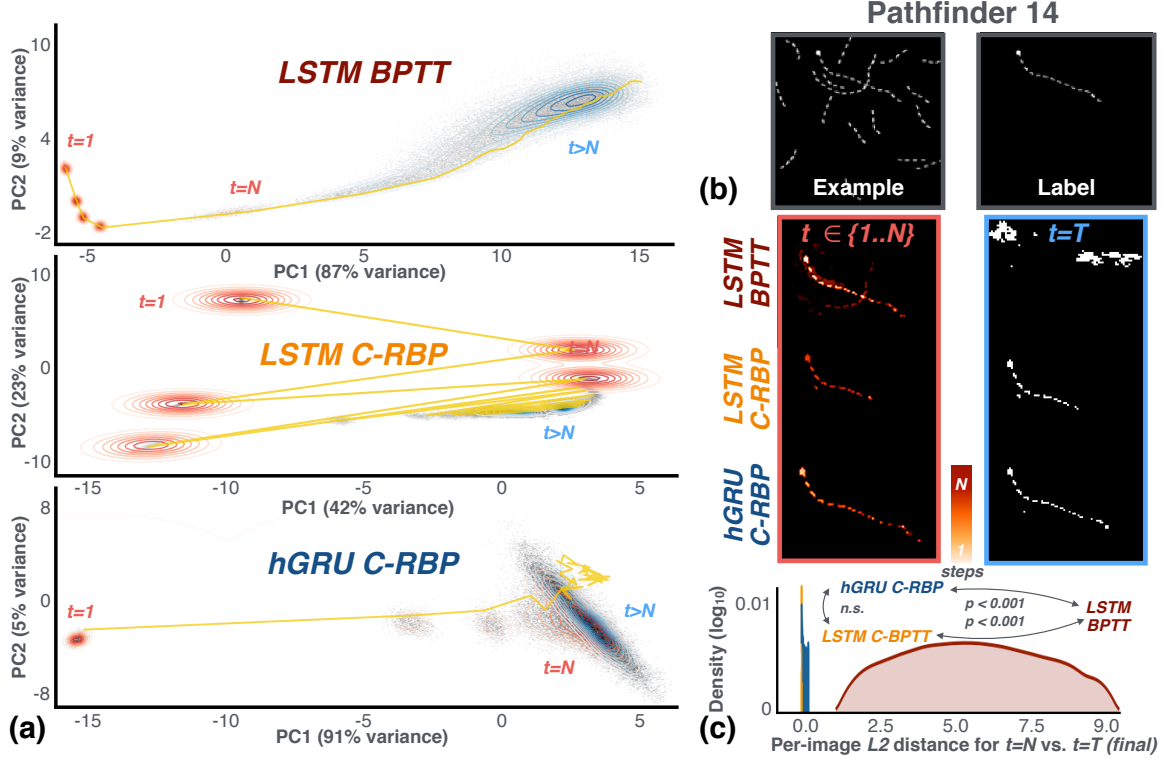


Figure 3.7: **Convolutional LSTMs trained with (BPTT) exhibit unstable dynamics, like the BPTT-trained hGRUs.** Once again, LCP corrects this pathology. (a) Visualization of convLSTM and hGRU state spaces following the state space method described in Section . Here, the BPTT-LSTM was trained for 6 steps, the **C-RBP LSTM** for 60 steps, and the **C-RBP hGRU** for 40 steps. Grey dots are the 2D-histogram of projected hidden states, red contours are hidden state densities up to the task-optimized N steps, and blue contours are hidden state densities beyond that point ($t > N$). Exemplar dynamics for a single image are plotted in yellow. While dynamics of the **BPTT** trained model diverge when $t > N$, models trained with LCP did not. (b) Model dynamics are reflected in their performance on Pathfinder-14 at $t = N$ and $t = T$ steps of processing. (c) Two-sample KS-tests indicate that the distance in state space between $t = N$ and $t = T$ hidden states is significantly greater for the BPTT-trained convLSTM than for either of the models trained with **C-RBP** (n.s. = not significant).

feedforward models on simple computer vision benchmarks like CIFAR.

Deep Equilibrium Models A recent extension to RBP includes a Neumann-series computation to approximate the inverse of a dynamical system’s Jacobian [27]. Separately, deep equilibrium models (DEQ) use root-finding algorithms to exploit an identical formulation as in Eq. 3 to compute the steady state h^* [28]. Both RBP and DEQ algorithms are effective for sequence modeling and meta-learning tasks, but have yet to be extended to vision. We attempted to train our hGRU models on Pathfinder 14 with DEQ, but it performed as poorly as the RBP-trained model while using more GPU memory and taking longer to train.

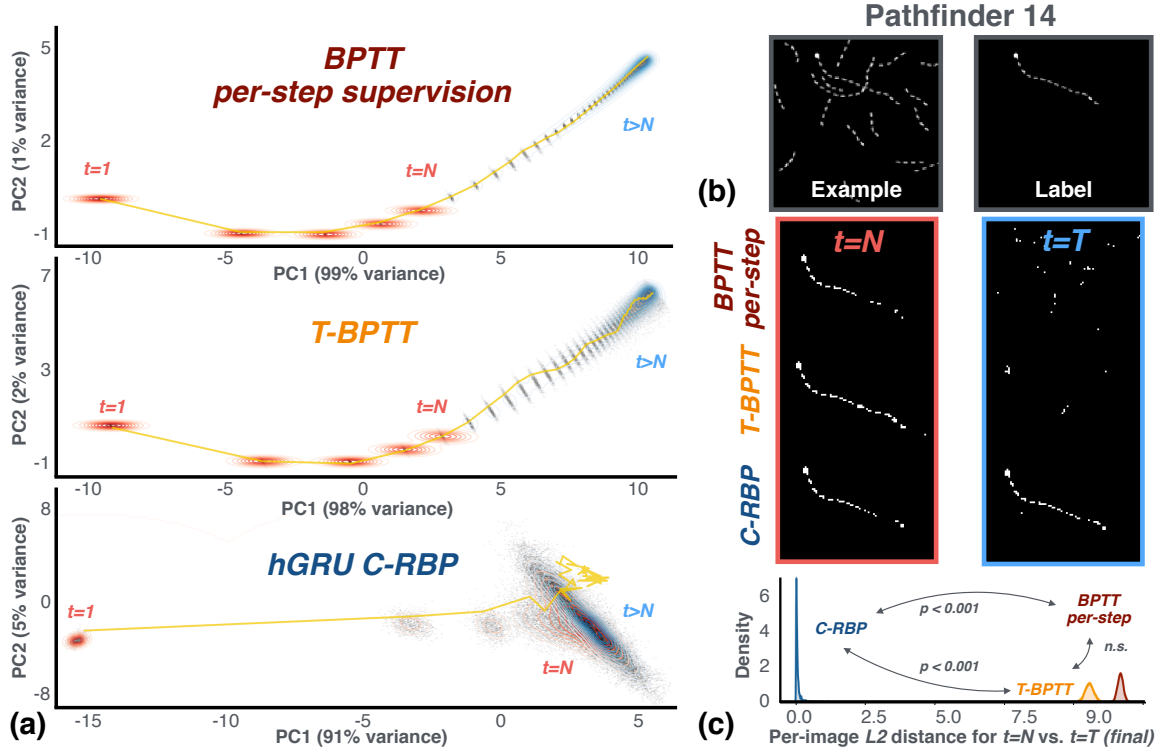


Figure 3.8: Additional state space analyses showed that alternatives to BPTT do not resolve the unstable dynamics we observed for recurrent CNNs. Here, **BPTT per-step supervision** refers to a model which was optimized with a loss evaluated on each of its 6 steps of processing. **T-BPTT** refers to a model trained with truncated backprop, for which gradients were accumulated over 3 steps of its 6 steps of processing. (a,b,c) These BPTT alternatives train models with unstable dynamics, which forgot task information after the optimized $t = N$ steps of processing. The distances between $t = N$ and $t = T$ hidden states are significantly greater for hGRUs trained with these algorithms than for an hGRU trained with **C-RBP** (n.s. = not significant).

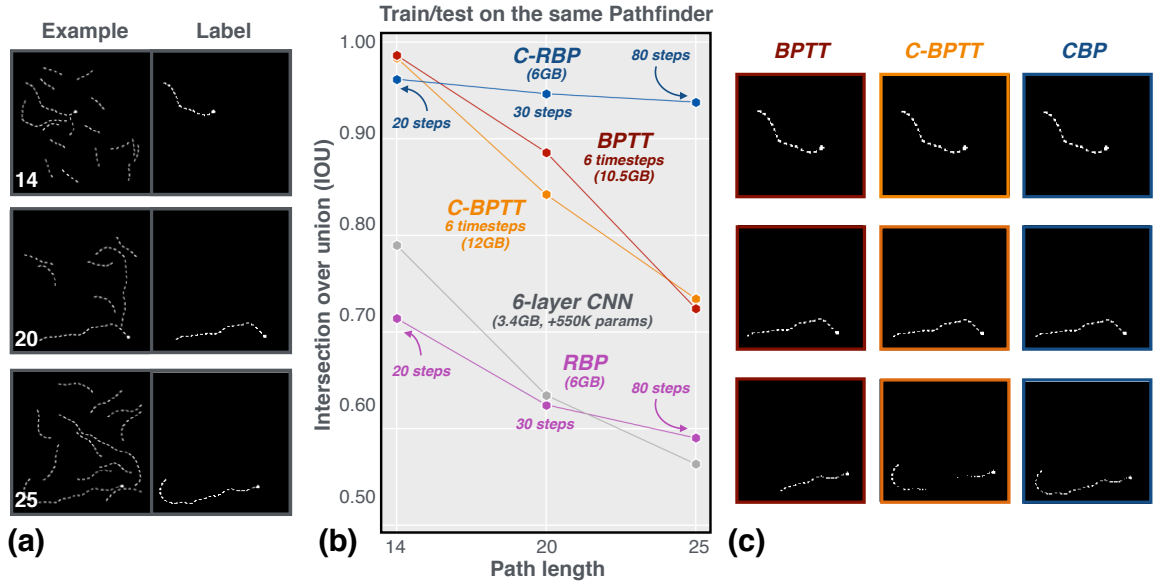


Figure 3.9: **Evaluating model performance.** Exemplars from the (a) *Pathfinder* challenge, along with (b) model performance on each of these datasets, and (c) predicted contours from hGRUs trained with the different algorithms.

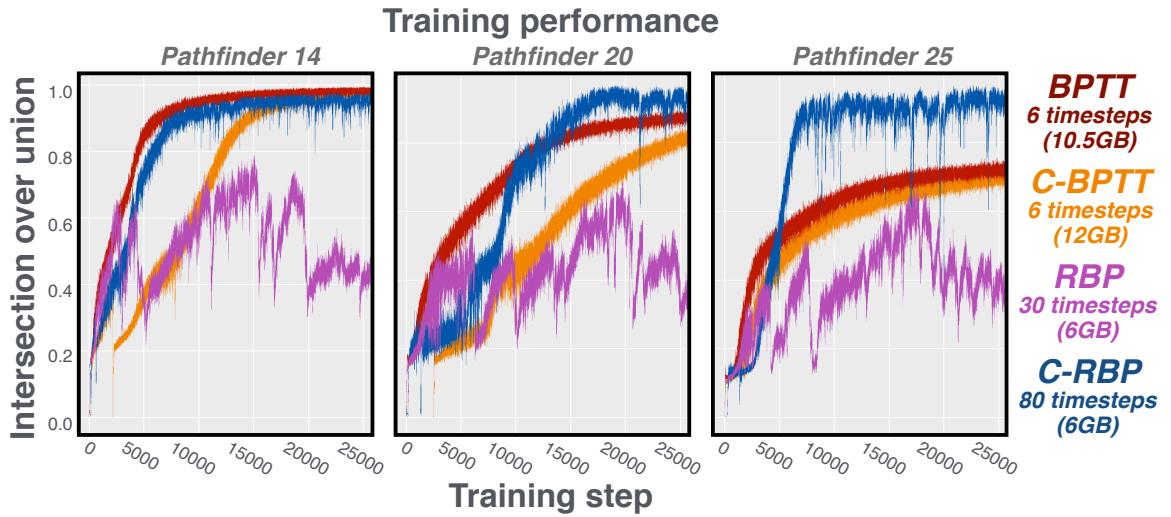


Figure 3.10: **Performance of hGRUs during training on *Pathfinder* challenge datasets.** The RBP-trained model struggles to fit any dataset, unlike the models trained with BPTT, C-BPTT, or C-RBP.

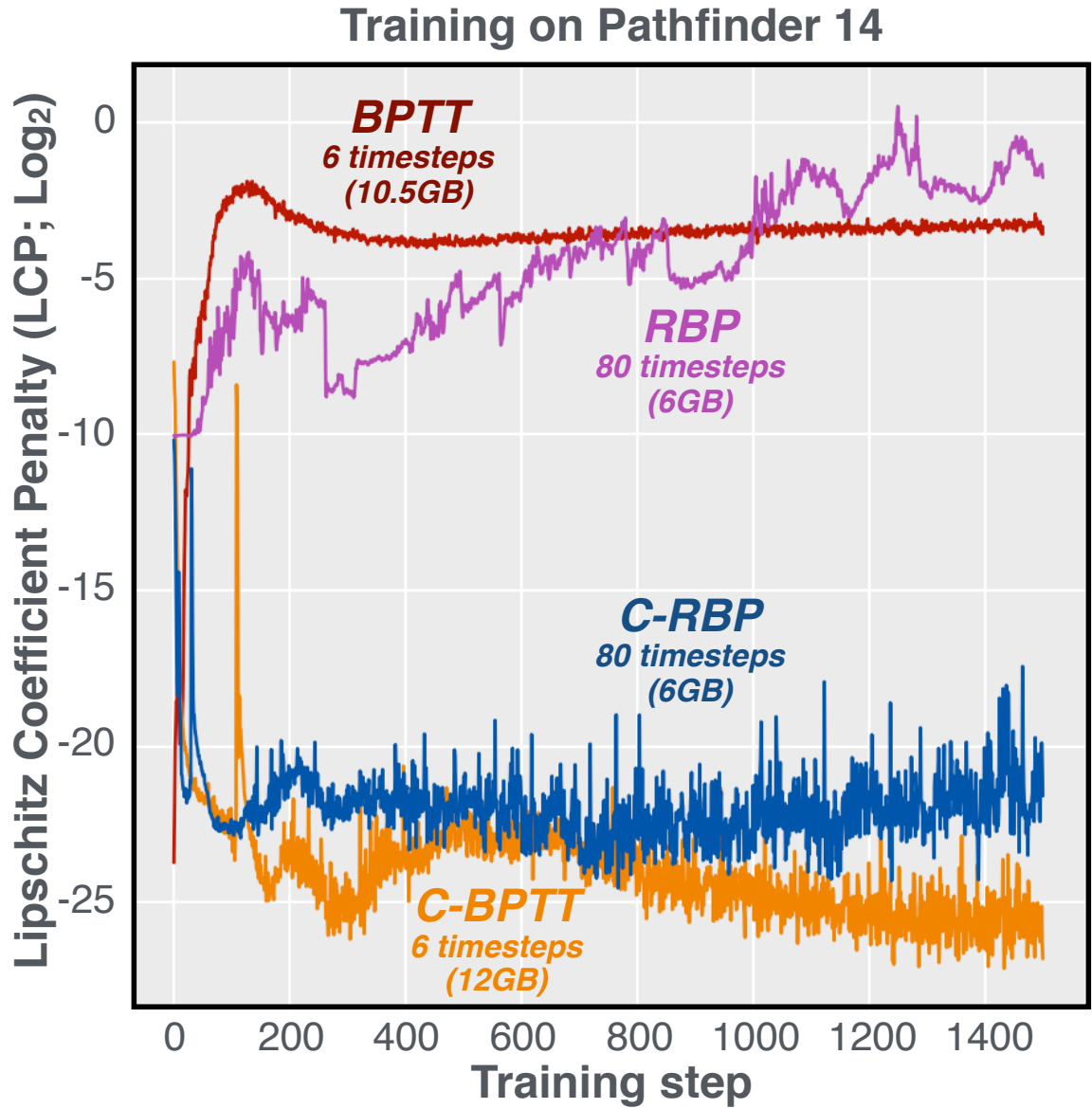


Figure 3.11: **Tracking LCP.** The value of our LCP (computed with Eq. 6 in the main text) over the course of training for models that minimize it (**C-RBP**, **C-BPTT**) and models that do not (**RBP**, **BPTT**). In other words, the magnitude of this correlates with the stability/instability of model dynamics.

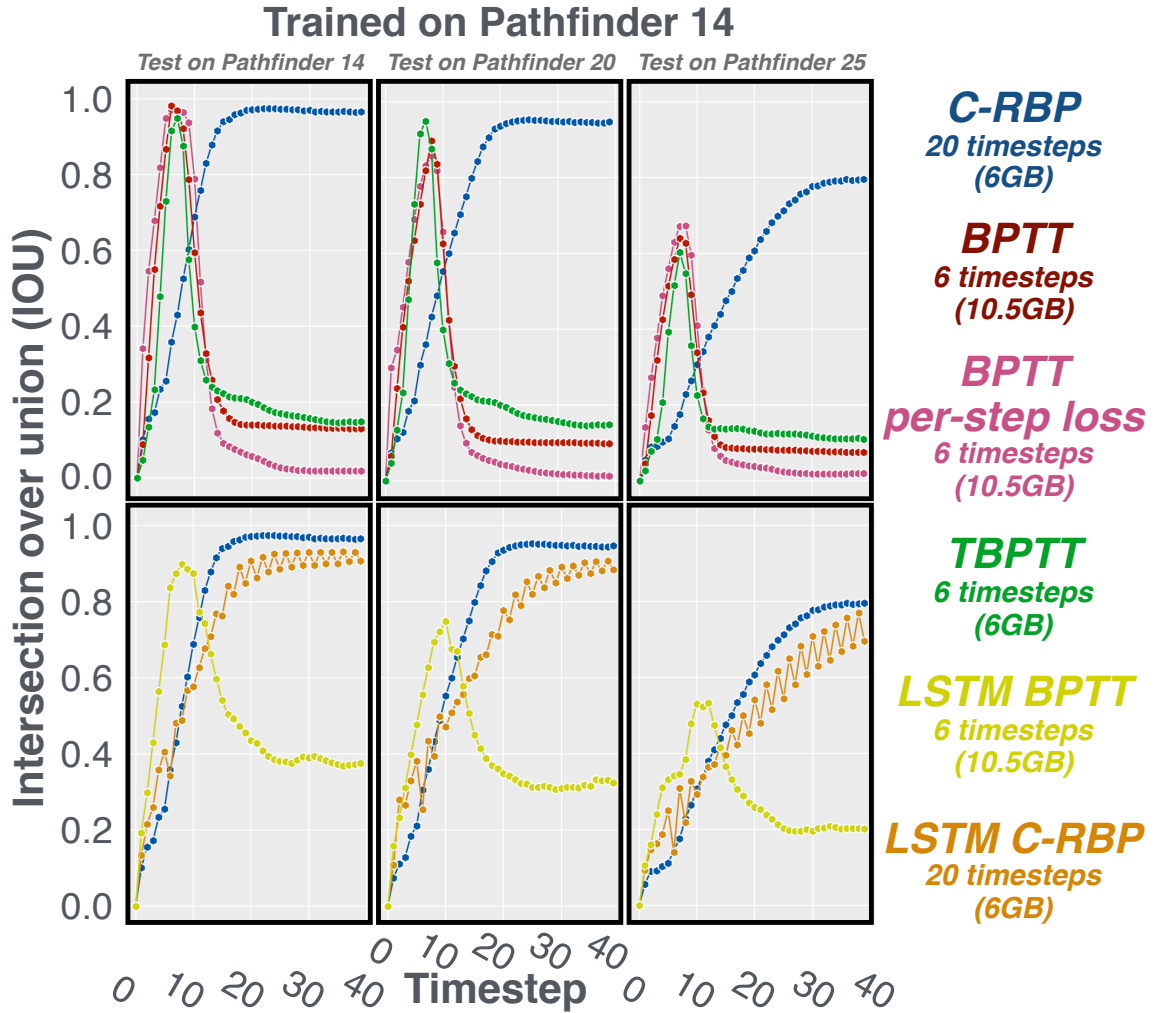


Figure 3.12: **Generalization performance for hGRUs and convLSTMs trained with BPTT and alternatives to BPTT.** Models were trained on Pathfinder 14 and tested on Pathfinder 14/20/25. For reference, performance of the hGRU trained with C-RBP is plotted in both rows. BPTT per-step loss means that a loss was computed on each of the 6 steps of hGRU training, and weights were optimized with BPTT. In contrast, a loss was only calculated on the final step for BPTT. TBPTT is truncated backprop through time, where gradients were computed over 3 steps of the 6-step dynamics. The LSTM trained with BPTT was trained for 6 steps, whereas the LSTM trained with C-RBP was trained for 60.

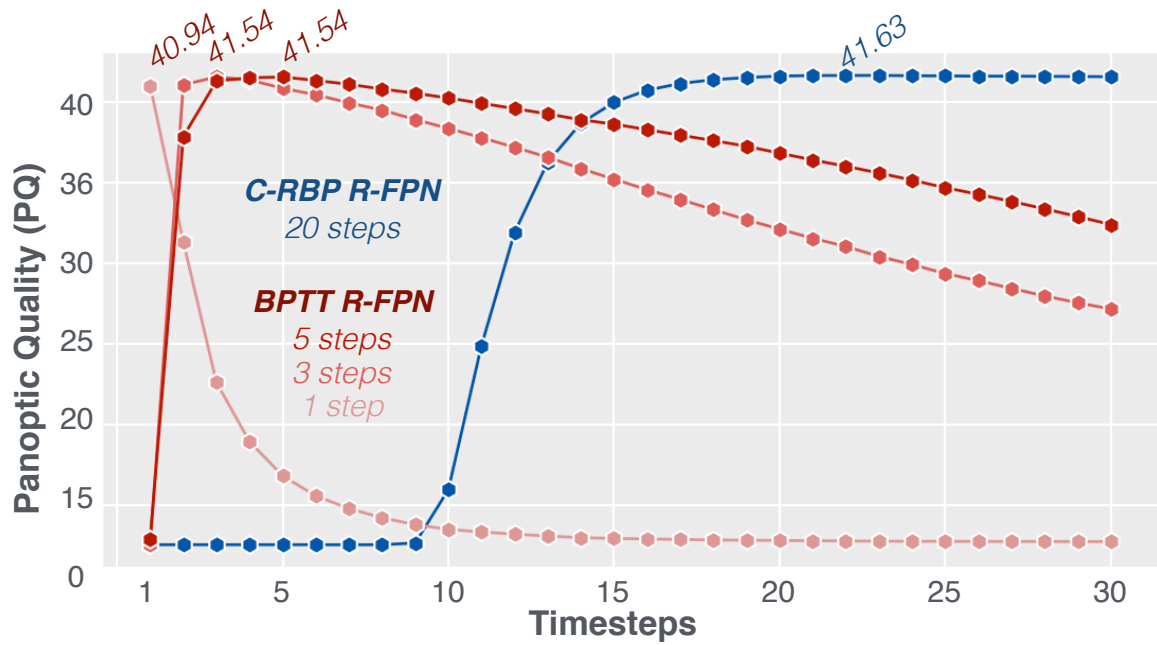


Figure 3.13: **Recurrent model performance on MSCOCO Panoptic Segmentation.** Performance was computed for each of 30 steps of processing for models trained with BPTT and C-RBP. The C-RBP models achieve better and more stable performance.

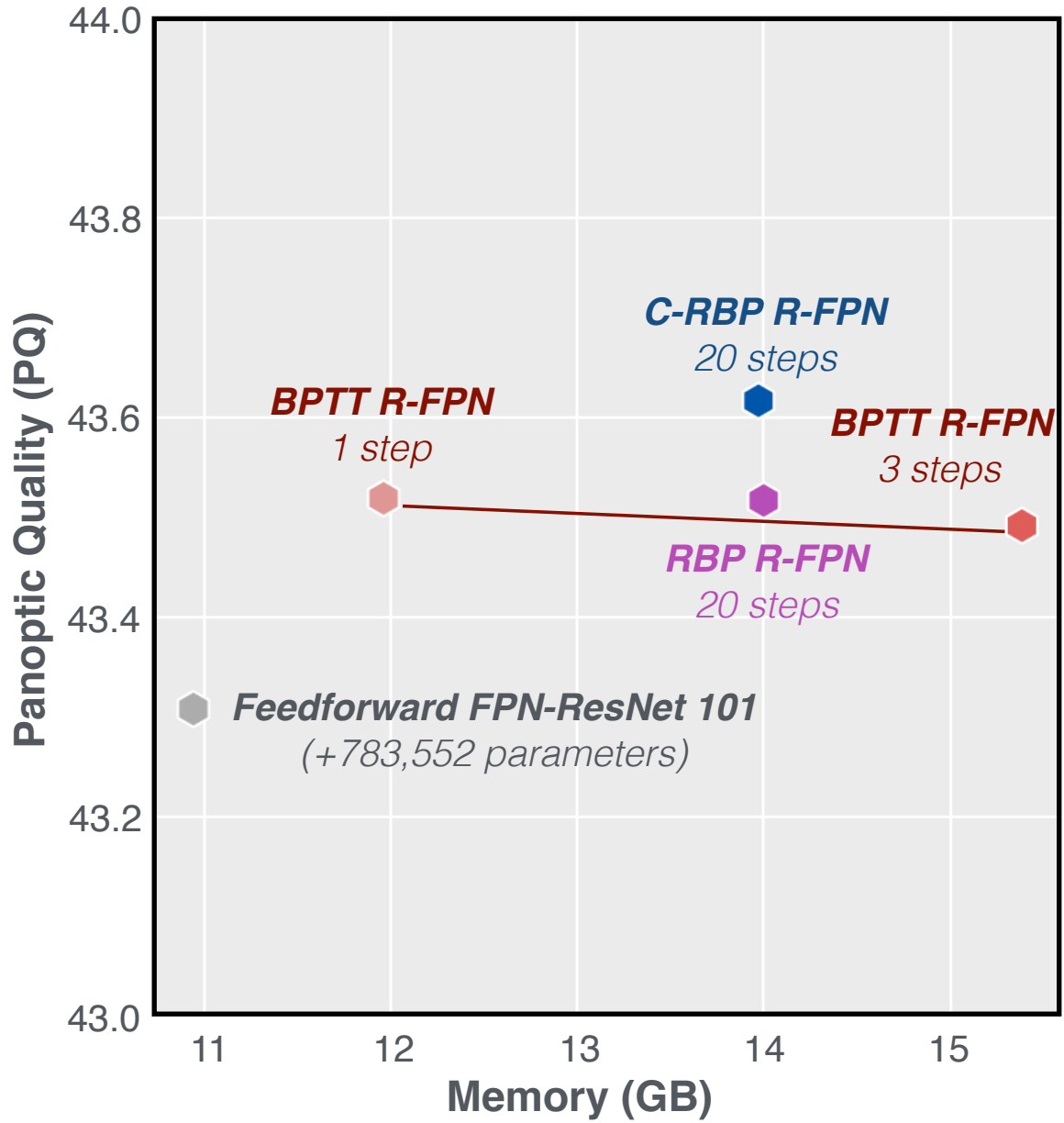


Figure 3.14: **Panoptic segmentation performance.** Our recurrent Panoptic Segmentation models also outperform the feedforward standard when both are given ResNet-101 backbones and the $3\times$ training schedule (see <https://bit.ly/dtcon> for details on this training routine). The C-RBP model, trained for 20 steps, outperforms any other tested version of the model.

Feedforward FPN-ResNet 50

C-RBP Recurrent FPN-ResNet 50

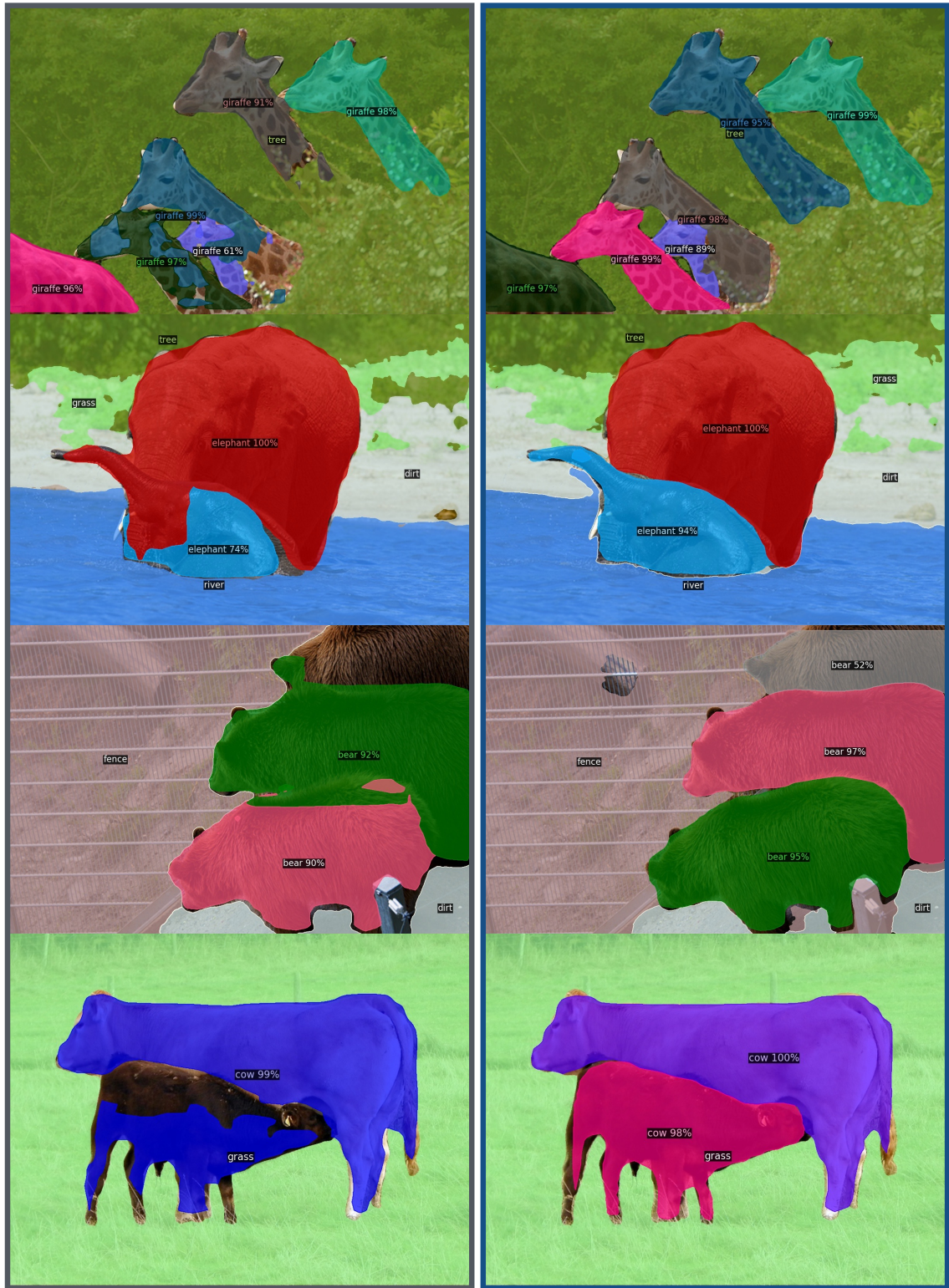


Figure 3.15: **Panoptic predictions** from the feedforward ResNet-50 FPN Mask-RCNN (left) and our recurrent version of the model trained with C-RBP (right).

Feedforward FPN-ResNet 50

C-RBP Recurrent FPN-ResNet 50

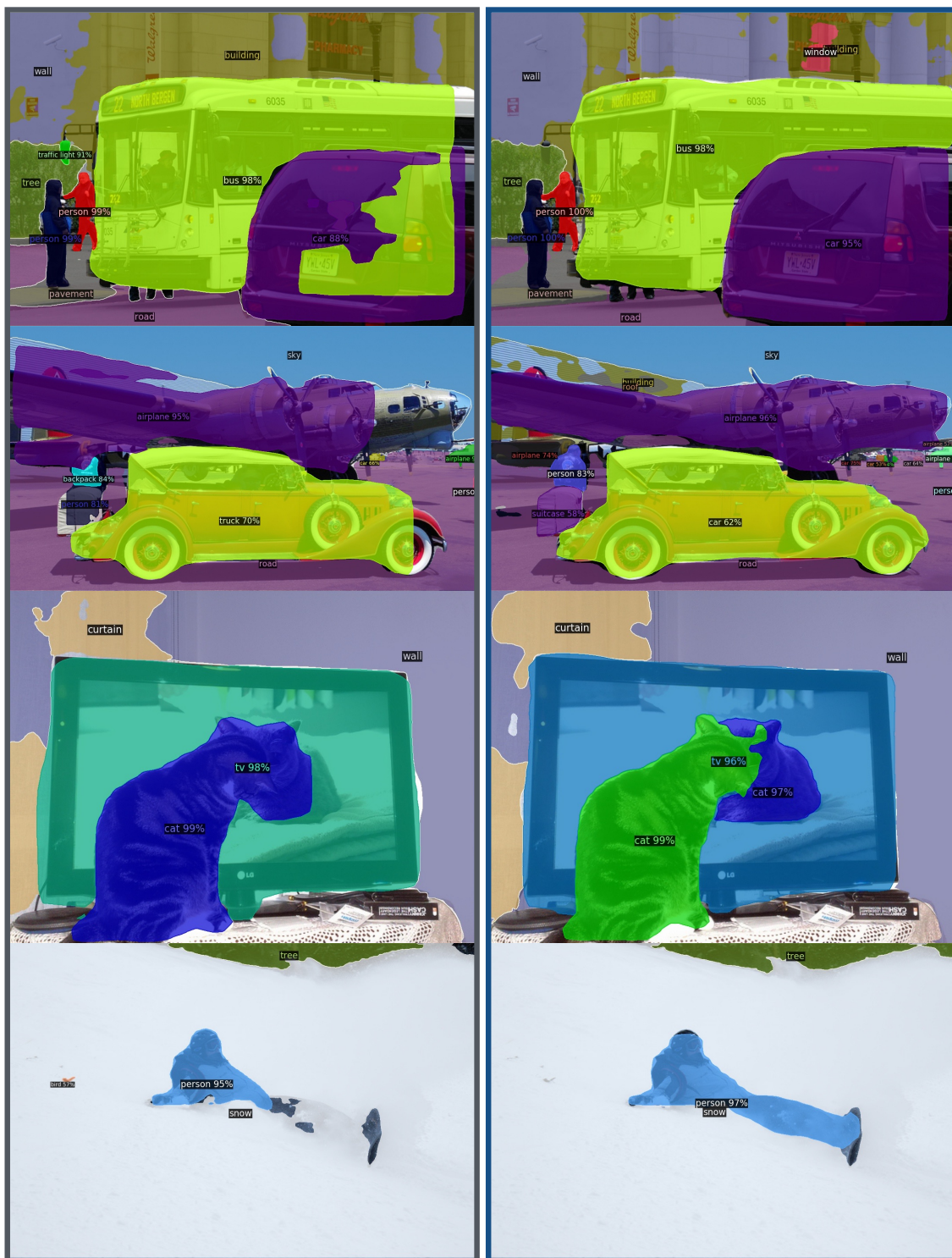


Figure 3.16: **Panoptic predictions** from the feedforward ResNet-50 FPN Mask-RCNN (left) and our recurrent version of the model trained with C-RBP (right).

3.11 References

- [1] P R Roelfsema, V A Lamme, and H Spekreijse. “The implementation of visual routines”. en. In: *Vision Res.* 40.10-12 (2000), pp. 1385–1411.
- [2] S Ullman. “Visual routines”. en. In: *Cognition* 18.1-3 (Dec. 1984), pp. 97–159.
- [3] Charles D Gilbert and Wu Li. “Top-down influences on visual processing”. In: *Nat. Rev. Neurosci.* 14.5 (May 2013), pp. 350–363.
- [4] Pieter R Roelfsema. “Cortical algorithms for perceptual grouping”. en. In: *Annu. Rev. Neurosci.* 29 (Jan. 2006), pp. 203–227.
- [5] James J DiCarlo and David D Cox. “Untangling invariant object recognition”. en. In: *Trends Cogn. Sci.* 11.8 (Aug. 2007), pp. 333–341.
- [6] Gabriel Kreiman and Thomas Serre. “Beyond the feedforward sweep: feedback computations in the visual cortex”. en. In: *Ann. N. Y. Acad. Sci.* 1464.1 (Mar. 2020), pp. 222–241.
- [7] Drew Linsley et al. “Learning long-range spatial dependencies with horizontal gated-recurrent units”. In: (May 2018). arXiv: 1805.08315 [cs.CV].
- [8] Junkyung Kim* et al. “Disentangling neural mechanisms for perceptual grouping”. In: *International Conference on Representation Learning* (2020).
- [9] Ming Liang and Xiaolin Hu. “Recurrent convolutional neural network for object recognition”. en. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, June 2015, pp. 3367–3375.
- [10] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. “Deeply-Recursive Convolutional Network for Image Super-Resolution”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, June 2016, pp. 1637–1645.
- [11] R Li et al. “Referring Image Segmentation via Recurrent Refinement Networks”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. June 2018, pp. 5745–5753.
- [12] Drew Linsley et al. “Learning long-range spatial dependencies with horizontal gated recurrent units”. In: *Advances in Neural Information Processing Systems 31*. Ed. by S Bengio et al. Curran Associates, Inc., 2018, pp. 152–164.
- [13] Hanlin Tang et al. “Recurrent computations for visual pattern completion”. en. In: *Proc. Natl. Acad. Sci. U. S. A.* 115.35 (Aug. 2018), pp. 8835–8840.
- [14] William Lotter, Gabriel Kreiman, and David Cox. “Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning”. In: (May 2016). arXiv: 1605.08104 [cs.LG].
- [15] Roos Houtkamp and Pieter R Roelfsema. “Parallel and serial grouping of image elements in visual perception”. en. In: *J. Exp. Psychol. Hum. Percept. Perform.* 36.6 (Dec. 2010), pp. 1443–1459.

- [16] Drew Linsley et al. “Robust neural circuit reconstruction from serial electron microscopy with convolutional recurrent networks”. In: (Nov. 2018). arXiv: 1811.11356 [cs.CV].
- [17] Drew Linsley et al. “Recurrent neural circuits for contour detection”. In: *International Conference on Learning Representations* (2020).
- [18] Jonas Kubilius et al. “CORnet: Modeling the Neural Mechanisms of Core Object Recognition”. en. Sept. 2018.
- [19] Jonas Kubilius et al. “Brain-Like Object Recognition with High-Performing Shallow Recurrent ANNs”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H Wallach et al. Curran Associates, Inc., 2019, pp. 12805–12816.
- [20] A Nayebi et al. “Task-Driven Convolutional Recurrent Models of the Visual System”. In: *Neural Information Processing Systems*. 2018.
- [21] Kohitij Kar et al. “Evidence that recurrent circuits are critical to the ventral stream’s execution of core object recognition behavior”. en. In: *Nat. Neurosci.* (Apr. 2019).
- [22] F Sinz et al. “Stimulus domain transfer in recurrent models for large scale cortical population prediction on video”. In: *Adv. Neural Inf. Process. Syst.* (2018).
- [23] T C Kietzmann, C J Spoerer, L Sörensen, et al. “Recurrence required to capture the dynamic computations of the human ventral visual stream”. In: *arXiv preprint arXiv* (2019).
- [24] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: (May 2014). arXiv: 1405.0312 [cs.CV].
- [25] P J Werbos. “Backpropagation through time: what it does and how to do it”. In: *Proc. IEEE* 78.10 (Oct. 1990), pp. 1550–1560.
- [26] Mingxing Tan and Quoc V Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: (May 2019). arXiv: 1905.11946 [cs.LG].
- [27] Renjie Liao et al. “Reviving and Improving Recurrent Back-Propagation”. In: (Mar. 2018). arXiv: 1803.06396 [cs.LG].
- [28] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. “Deep Equilibrium Models”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [29] L B Almeida. “A Learning Rule for Asynchronous Perceptrons with Feedback in a Combinatorial Environment”. In: *Proceedings of the IEEE First International Conference on Neural Networks San Diego, CA*. Ed. by M Caudil and C Butler. 1987, pp. 609–618.
- [30] F J Pineda. “Generalization of back-propagation to recurrent neural networks”. en. In: *Phys. Rev. Lett.* 59.19 (Nov. 1987), pp. 2229–2232.
- [31] J J Hopfield. “Neural networks and physical systems with emergent collective computational abilities”. en. In: *Proc. Natl. Acad. Sci. U. S. A.* 79.8 (Apr. 1982), pp. 2554–2558.
- [32] Walter Rudin. *Principles of mathematical analysis*. Vol. 3. McGraw-Hill New York, 1964.

- [33] Franco Scarselli et al. “The graph neural network model”. en. In: *IEEE Trans. Neural Netw.* 20.1 (Jan. 2009), pp. 61–80.
- [34] Sergey Ioffe and Christian Szegedy. “Batch normalization: accelerating deep network training by reducing internal covariate shift”. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. JMLR.org, July 2015, pp. 448–456.
- [35] Diederik P Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: (Dec. 2014). arXiv: 1412.6980 [cs.LG].
- [36] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep Sparse Rectifier Neural Networks”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, 2011, pp. 315–323.
- [37] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterton. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010, pp. 249–256.
- [38] Caglar Gulcehre et al. “Noisy activation functions”. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML’16. New York, NY, USA: JMLR.org, June 2016, pp. 3059–3068.
- [39] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)”. In: *International Conference on Learning Representations* (Nov. 2015).
- [40] John Miller and Moritz Hardt. “Stable Recurrent Models”. In: *International Conference on Representation Learning (ICLR)* (Sept. 2018).
- [41] A Kirillov et al. “Panoptic Segmentation”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019, pp. 9396–9405.
- [42] Danique Jeurissen, Matthew W Self, and Pieter R Roelfsema. “Serial grouping of 2D-image regions with object-based attention in humans”. In: *Elife* 5 (June 2016), e14320.
- [43] Jeffrey Dean et al. “Large Scale Distributed Deep Networks”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F Pereira et al. Curran Associates, Inc., 2012, pp. 1223–1231.
- [44] Minsoo Rhu et al. “vDNN: virtualized deep neural networks for scalable, memory-efficient neural network design”. In: *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO-49 Article 18. Taipei, Taiwan: IEEE Press, Oct. 2016, pp. 1–13.
- [45] Benjamin Dauvergne and Laurent Hascoët. “The Data-Flow Equations of Checkpointing in Reverse Automatic Differentiation”. In: *Computational Science – ICCS 2006*. Springer Berlin Heidelberg, 2006, pp. 566–573.

- [46] Jianxu Chen et al. “Combining Fully Convolutional and Recurrent Neural Networks for 3D Biomedical Image Segmentation”. In: *Advances in Neural Information Processing Systems 29*. Ed. by D D Lee et al. Curran Associates, Inc., 2016, pp. 3036–3044.
- [47] Audrūnas Gruslys et al. “Memory-efficient backpropagation through time”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS’16. Barcelona, Spain: Curran Associates Inc., Dec. 2016, pp. 4132–4140.
- [48] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. “Augmented Neural ODEs”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H Wallach et al. Curran Associates, Inc., 2019, pp. 3140–3150.
- [49] Y Bengio, P Simard, and P Frasconi. “Learning long-term dependencies with gradient descent is difficult”. en. In: *IEEE Trans. Neural Netw.* 5.2 (1994), pp. 157–166.
- [50] Y Yoshida and T Miyato. “Spectral norm regularization for improving the generalizability of deep learning”. In: *arXiv preprint arXiv:1705.10941* (2017).
- [51] T Miyato et al. “Spectral normalization for generative adversarial networks”. In: *International Conference on Learning Representations* (2018).
- [52] Shixiang Gu and Luca Rigazio. “Towards Deep Neural Network Architectures Robust to Adversarial Examples”. In: (Dec. 2014). arXiv: 1412.5068 [cs.LG].
- [53] Ishaan Gulrajani et al. “Improved training of wasserstein GANs”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., Dec. 2017, pp. 5769–5779.
- [54] Xiang Wei et al. “Improving the Improved Training of Wasserstein GANs: A Consistency Term and Its Dual Effect”. In: *International Conference on Learning Representations* (Mar. 2018). arXiv: 1803.01541 [cs.CV].
- [55] Tim Cooijmans et al. “Recurrent Batch Normalization”. In: *International Conference on Learning Representations*. 2017.
- [56] Niru Maheswaranathan et al. “Reverse engineering recurrent networks for sentiment classification reveals line attractor dynamics”. In: (June 2019). arXiv: 1906.10720 [cs.LG].
- [57] T Lin et al. “Feature Pyramid Networks for Object Detection”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017, pp. 936–944.
- [58] Yuxin Wu and Kaiming He. “Group Normalization”. In: (Mar. 2018). arXiv: 1803.08494 [cs.CV].
- [59] Dileep George et al. “A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs”. en. In: *Science* 358.6368 (Dec. 2017).
- [60] Courtney J Spoerer, Patrick McClure, and Nikolaus Kriegeskorte. “Recurrent Convolutional Neural Networks: A Better Model of Biological Object Recognition”. en. In: *Front. Psychol.* 8 (Sept. 2017), p. 1551.

- [61] Amir R Zamir et al. “Feedback Networks”. In: (Dec. 2016). arXiv: 1612.09508 [cs.CV].
- [62] Haiguang Wen et al. “Deep Predictive Coding Network for Object Recognition”. In: (Feb. 2018). arXiv: 1802.04762 [cs.CV].
- [63] Qianli Liao and Tomaso Poggio. “Bridging the Gaps Between Residual Learning, Recurrent Neural Networks and Visual Cortex”. In: (Apr. 2016). arXiv: 1604.03640 [cs.LG].
- [64] Salah Rifai et al. “Contractive auto-encoders: explicit invariance during feature extraction”. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. ICML’11. Bellevue, Washington, USA: Omnipress, June 2011, pp. 833–840.
- [65] S Hochreiter and J Schmidhuber. “Long short-term memory”. en. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780.
- [66] Patrice Y Simard, Mary B Ottaway, and Dana H Ballard. “Fixed Point Analysis for Recurrent Networks”. In: *Advances in Neural Information Processing Systems 1*. Ed. by D S Touretzky. Morgan-Kaufmann, 1989, pp. 149–159.
- [67] Jiong Zhang, Qi Lei, and Inderjit S Dhillon. “Stabilizing Gradients for Deep Neural Networks via Efficient SVD Parameterization”. In: (Mar. 2018). arXiv: 1803.09327 [cs.LG].
- [68] Wanli Chen et al. “Prostate Segmentation using 2D Bridged U-net”. In: (July 2018). arXiv: 1807.04459 [cs.CV].

Chapter 4

The Utility of Visual Representation Dynamics in Deep RNNs

4.1 Synopsis

In Chapter 3, we made a case for the versatility of deep recurrent vision models. Specifically, we introduced a biologically-inspired recurrent feedback model, the horizontal gated recurrent unit (hGRU), and proposed a learning algorithm for RNNs motivated by attractor dynamics principles that stabilized model dynamics and made it practically amenable without compromising on expressivity. We briefly went over the possibility that stable model dynamics could support the emergence of several interesting phenomena, with one of them being a flood-filling strategy for panoptic segmentation. In this chapter, we seek to explore this question further. Specifically, we are interested in quantifying how visual representational dynamics shaped by feedback mechanisms support the downstream utilization for tasks that are at the intersection of visual perception and cognition. Keeping this theme in mind, we break this chapter down into three independent case studies. In the first study, we draw parallels between the time course of our model’s dynamics and that of human decision-making in an incremental grouping task. In the second study, we propose that feedback mechanisms are critical for visual simulation, a core property of visual intelligence that emerges from our model dynamics. In the final case study, we scrutinize the performance and behavior of deep reinforcement learning agents and conclude that stable and structured visual representations enable efficient policy learning in tasks with a high degree of visual demand.

4.2 Use case I: Modeling human behavior in a serial grouping task

We reviewed in Chapter 3 the ability of our feedback model (hGRU), trained with constraints from attractor dynamics, to learn long-range spatial dependencies efficiently. Specifically, we saw that C-RBP optimized models learn to implement a visual routine that resembles a flood-fill algorithm, despite no explicit supervision to do so. This particular visual routine is reminiscent of mechanisms speculated to underpin our visual system’s perceptual organization process. Primate visual systems are thought to extract global structure in scenes by sequentially stitching together primitive visual features in a process commonly known as “perceptual grouping”. Furthermore, psychophysics studies measuring reaction times of human participants in perceptual grouping tasks posit that the speed of human decisions is modulated by object-based attention [1–3]. While mechanistic models of attention spreading can account for human reaction time data, there is a paucity of computational models that can reconcile neural implementation, neural dynamics, and behavioral findings.

Incremental grouping, a special form of the more general perceptual grouping process, is believed to be slow, serial, and reliant on local- and long-range feedback [3, 4]. We argue that our feedback model, the C-RBP optimized hGRU, offers the perfect scaffold to study incremental grouping. First, the hGRU circuit has the necessary implementational substrates. Second, with the C-RBP constraint, its representational dynamics can offer a bridge between model and human strategies. Third, the circuit can be optimized for task performance on large-scale benchmarks, thus linking algorithmic explanations to behavior.

4.2.1 Methods

The COCO-Dots dataset

Toward our goal of understanding the parallels between the visual routines discovered by our feedback model and the algorithms used by human observers, we created a visual challenge that mimicked a task paradigm used in human behavioral experiments. In [1], participants were asked to judge if a cue dot fell on the same or different object silhouette than a fixation dot. We recreate this task, on naturalistic image contours, in our dataset COCO-Dots. Each sample in COCO-Dots ($N \sim 250\text{K}$) consisted of one or more object silhouettes derived from the Microsoft-COCO dataset [5], a large-scale visual recognition benchmark. Samples $x_{1..N} \in \mathbb{R}^{150 \times 150 \times 3}$ were three channel tensors, with the first channel containing the object outlines, the second channel a “cue” dot, and the third channel a “fixation” dot. x_j was deemed a positive sample with class label $y_j = 1$ if both the cue and fixation dots lay on the same object silhouette, and a negative sample with $y_j = 0$ otherwise. The positions of the cue and fixation dots were randomized such that (1) there was an equal number of positive and negative samples, and (2) the distance between the cue and fixation dots was not a predictor of class label.

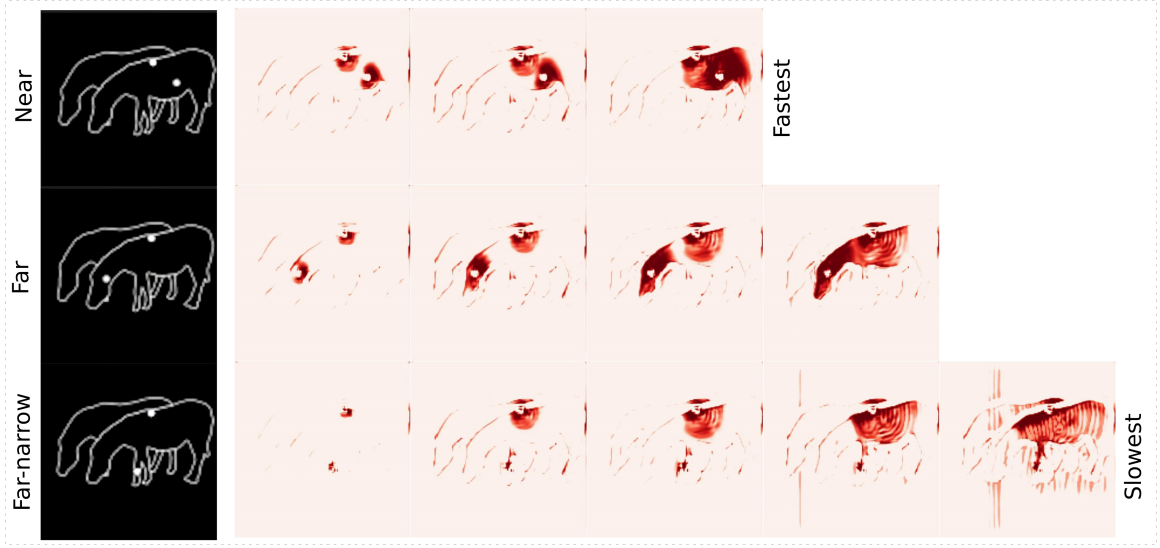


Figure 4.1: **Activity dynamics of an hGRU trained to solve the COCO-Dots challenge recapitulate human decision-making time courses.** We observe that the hGRU’s internal activity dynamics are a function of the cue and fixation dot configuration. By considering the first timepoint at which activations from the cue and fixation dots merge as a proxy for the model reaction time, we can reliably correlate the time courses of model behavior and human decision-making. In agreement with human psychophysics data presented in Jeurissen et al. [1], we find that our models are fast when the two dots are close to each other and slow when they are either far apart or one of them resides in a narrow region in space.

Network models

We trained our feedback model, hGRU, with the learning algorithm C-RBP (described in Chapter. 3) to perform a classification task on the COCO-Dots challenge. All models were instantiated with horizontal kernels of size 5×5 and were trained for 20 epochs with a batch size of 128. Model dynamics were unrolled for $T = 80$ time steps. The training objective was to minimize a Cross-Entropy loss, and model parameters were updated using the Stochastic Gradient Descent algorithm implemented via an Adam optimizer ($\text{lr}=1e-3$) [6].

4.2.2 Results

Classification performance

Our models successfully learn to solve the COCO-Dots classification challenge with a mean validation accuracy of 98%. Moreover, zero-shot generalization to stimuli used in [1] yielded an accuracy of 95%.

Analyzing model dynamics

Visualizing the internal dynamics of an hGRU trained on the COCO-Dots challenge revealed that the model learned a flood-filling strategy (similar to [7]) despite task performance being the only constraint applied during training. We further sought to analyze if the dynamical patterns of activity seen in the model reflect, to some degree, the decision-making process of human observers. To do so, we first needed a reliable metric for what would constitute a model *reaction time*. Deriving reaction time measures from neural dynamics is a challenging and open research problem [8]. Here, we use a rudimentary strategy and consider the first timepoint at which activations from the cue and fixation dots merge as a proxy for the model reaction time (RT). In Fig. 4.1, we illustrate how the internal dynamics of our model evolve with time as a function of the cue and fixation dot positions.

In the set of $n = 22$ stimuli on which we had human behavioral data, we found that the model exhibited a similar pattern of reaction times as humans, indicating that its circuit constraints reflect possible neural substrates for the visual routines of incremental grouping in humans. Specifically, the hGRU takes longer to reach a decision when human RT is high $r = .40$. Model RT was higher when the cue is far versus close to fixation ($t(42.18) = 3.37, p < .01$) (Fig. 4.1 middle row), and when it is equally far, but on a narrower part of the object ($t(43.45) = 2.82, p < .05$) (Fig. 4.1 bottom row). Finally, the correlation between RTs from our model and an ideal mechanistic model (Growth-cone [2]) distance measure was .67, versus .54 for simply the Euclidean distance between the cue and fixation dots. These results mimic the behavioral findings from Jeurissen, Self, and Roelfsema, lending credence to our learning-based approach for modeling incremental grouping processes.

4.3 Use case II: Studying the mechanisms underlying visual simulation

Humans and non-human primates display a remarkable ability to make rapid visual inferences, even when sensory inputs are impoverished. Think of the times when you have pictured yourself finally achieving that 225lb bench press. This is an example of you imagining future states in your environment using a constructed mental model. This process, known as mental simulation, is posited to underlie our ability to reason about and interact effectively with our environments [9–12]. Visual simulation, an instantiation of the more general mental simulation variety, is one that specifically relies on visual perceptual representations. One of the hallmarks of visual simulation is that its underlying cognitive mechanisms do not seem to implement a veridical algorithm. Human judgments in physical reasoning tasks more closely resemble approximate solutions that embody uncertainty than exact physics-based algorithms [11]. Despite the bulk of behavioral evidence for theories of visual simulation, computational-level hypotheses are still lacking. For instance, the relative contributions of feedforward and feedback mechanisms for visual imagery are unspecified [13, 14]. Here, we build an image-computable feedback model and train it on a task that putatively involves visual simulation. We explore the visual representational dynamics of this model, only constrained by task performance, and seek to understand if simulation-like strategies emerge in its solution space.

4.3.1 Methods

The *Planko* challenge

Inspired by prior work in visual simulation that measured primate accuracy in simulating the trajectory of moving balls and used fMRI to identify regions of the cortex that correlated with their behavior [15], we develop the *Planko* task. Each *Planko* task board depicts a ball at the top of the screen placed above ten randomly oriented and positioned planks. A catcher is placed at the bottom of the screen (Fig. 4.2a). Each plank is parameterized by its angle of inclination, length, and its position on the screen. The physics of this world are specified by Newton Dynamics (<http://www.newtondynamics.com>). The ball’s trajectory as it falls down is tracked to determine if it eventually lands in the catcher, in which case it is considered a positive example or falls to the ground, in which case it is considered a negative example. We systematically control for the challenge associated with solving *Planko* boards by controlling the range of variation in each of its elements.

Network models

We train a suite of neural network models to classify each *Planko* game board into one of either a positive or a negative class. The overall performance on this task was the *only* constraint these models were subjected to. Model parameters were optimized with Stochastic Gradient Descent implemented via the Adam algorithm [16] with an initial learning rate of $3e - 4$. Binary Cross Entropy (BCE) was used as the training objective. Each train (test) dataset consisted of 200K (5K) *Planko* boards of

dimensions 64×64 pixels. The training was carried out on an NVIDIA TITAN Xp GPU for 100 epochs while measuring validation accuracy after each epoch over a held-out set of 10K boards.

As our feedback model, we implement the Index-and-Track (InT) circuit that is a variant of the canonical hGRU models used in Chapter. 3. We refer the reader to [17] for a complete description of InT. In addition to the InT, we also trained a simple feedforward 2-layer convolutional neural network (termed “Baseline CNN”), a standard VGG16 [18], and a 12-layer CNN with a parameter count identical to the InT circuit.

4.3.2 Results

Classification results

Analyzing the performance landscape of the suite of models we implemented, we observe that while both the feedforward and feedback models successfully solve easier versions of *Planko*, the feedback models were significantly better at solving the more challenging versions.

Decoding analysis

In addition to scrutinizing model performance, we also perform a decoding analysis to determine if the “imagined” positions of the ball are encoded in the visual representational dynamics of the InT circuit. Specifically, we train a decoder to predict ball positions from the InT model’s latent activity space (Fig. 4.2b) and minimize the mean squared error between these predictions and ground truth ball positions obtained from the physics simulator. Though not perfect, we find a reasonable degree of alignment between the imagined ball trajectories and physical paths that the ball would have taken (Fig. 4.2c), suggesting that the feedback model had learned to approximate the physics of this world.

4.3.3 Conclusion

We provide one of the first computational-level hypotheses for how visual simulation algorithms can be implemented by feedback model dynamics. We highlight that only our most performant model, on our most variable task, learns to “simulate” as an optimal strategy. Information about future states encoded in our model’s internal dynamics were non-veridical approximations, consistent with what has been reported in the literature. While this work is preliminary, we hope it paves the way for RNN-guided electrophysiology research to identify the neural substrates of mental simulation.

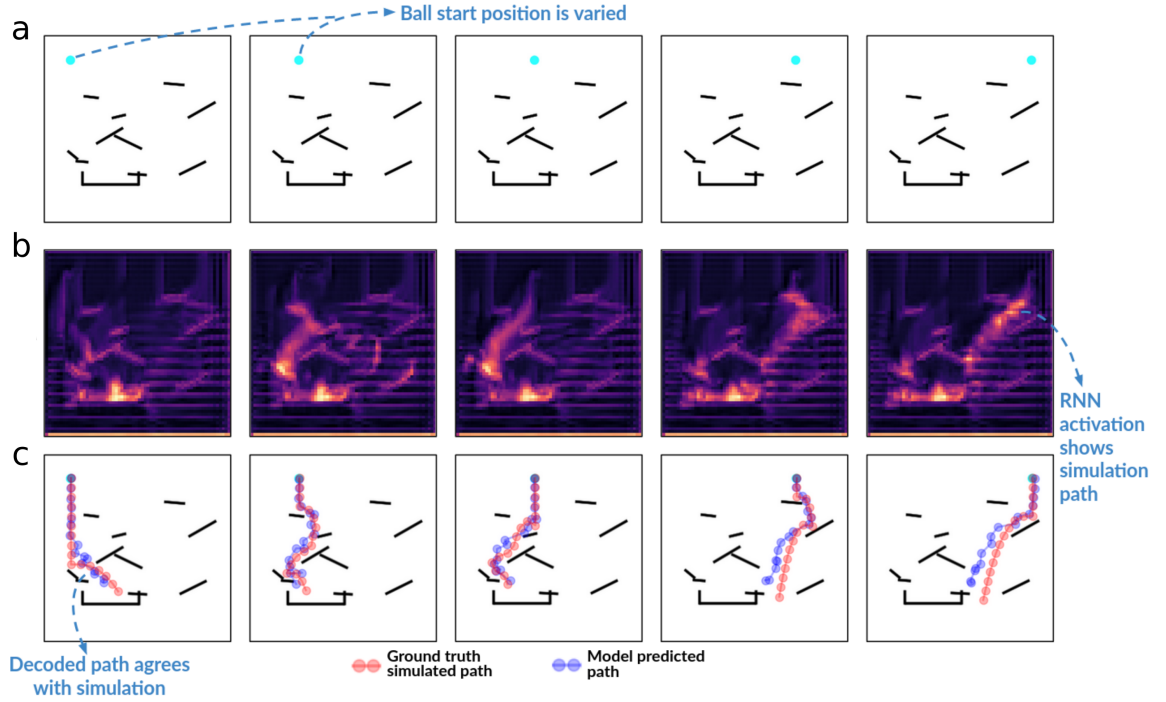


Figure 4.2: **Visual representation dynamics in our feedback model, InT, reveal a simulation strategy.** (a) Exempler inputs to the InT circuit. For illustration purposes, here we hold all components of a *Planko* board constant while only manipulating the ball position. (b) Visualizing the internal dynamics of the InT model right before it makes a decision reveals an evidence accumulation process. (c) “Simulated” ball trajectories can be decoded from the latent activity space of the feedback model. Visual simulation emerges as an optimal solution in this case, as the feedback circuit was never explicitly guided to perform this computation.

4.4 Use case III: Efficient decision-making in reinforcement learning agents

4.4.1 Introduction

Gibson famously argued that “The function of vision is not to solve the inverse problem and reconstruct a veridical description of the physical world. [... It] is to keep perceivers in contact with behaviorally relevant properties of the world they inhabit” (reviewed in [19]). The field of deep reinforcement learning (dRL) has followed Gibson’s tenet since the seminal introduction of deep Q-networks (DQN) [20]. DQNs rely on reward feedback to train their policies and perceptual systems at the same time to learn to play games from a *tabula rasa*. This end-to-end approach of training on individual environments and tasks has supported steady progress in the field of dRL, and newer reinforcement learning algorithms have yielded agents that achieve human or super-human performance in a variety of challenges – from Chess to Go and from Atari games to Starcraft [20–23]. But Gibson also argued that the ecological niche of animals allows them to exploit task-agnostic mechanisms to simplify the perceptual or behavioral demands of important tasks, like how humans rely on optic flow for navigation [19]. In the decades since Gibson’s writings, it has been found that humans can efficiently find or learn bespoke perceptual features that aid performance on a single task [24–27], or they can exploit previously learned generalist representations and task abstractions that are useful across multiple tasks and environments [27–31]. While there have been attempts at building similarly flexible dRL agents through meta-reinforcement learning [32–38], these approaches ignore the complexities of perceptual learning and carry large computational burdens that limit them to simplistic scenarios. There is a pressing need for approaches to training dRL agents that can meet the computational demands of a wide variety of environments and tasks.

One way to build generalist agents is to first reliably diagnose where the computational challenges of a given environment and task lie and adjust the agent to those demands. Is the perceptual challenge onerous? Is the reward signal for credit assignment especially sparse? Even partial answers to these questions are instructive for improving an agent, for instance, by pre-determining the extent to which it relies on feedback from the world to tune its policy and perception versus drawing from previously learned representations and task abstractions. The introduction of diverse video game challenges for dRL, such as the *Procgen* Benchmark [39], can serve as a starting point for this investigation. For example, take the game “Plunder” from *Procgen* (Figure 4.3a). Plunder has simple gameplay rules but poses a visual challenge: an agent is asked to shoot all objects that look like a provided cue. The difficulty here lies in assessing whether each object in the environment is the same or different than the cue; a visual routine that is difficult to learn for neural networks [40, 41]. For this reason, an agent that can draw from prior experience in learning relevant perceptual routines may perform better than one with a perceptual system tuned for this specific task from scratch. In contrast, the objects and environments of a game like “Leaper” (Figure 4.3a) can be identified by color. The game asks agents to use those objects to get from one side of the screen to the other, a task that can likely be learned from reward feedback.

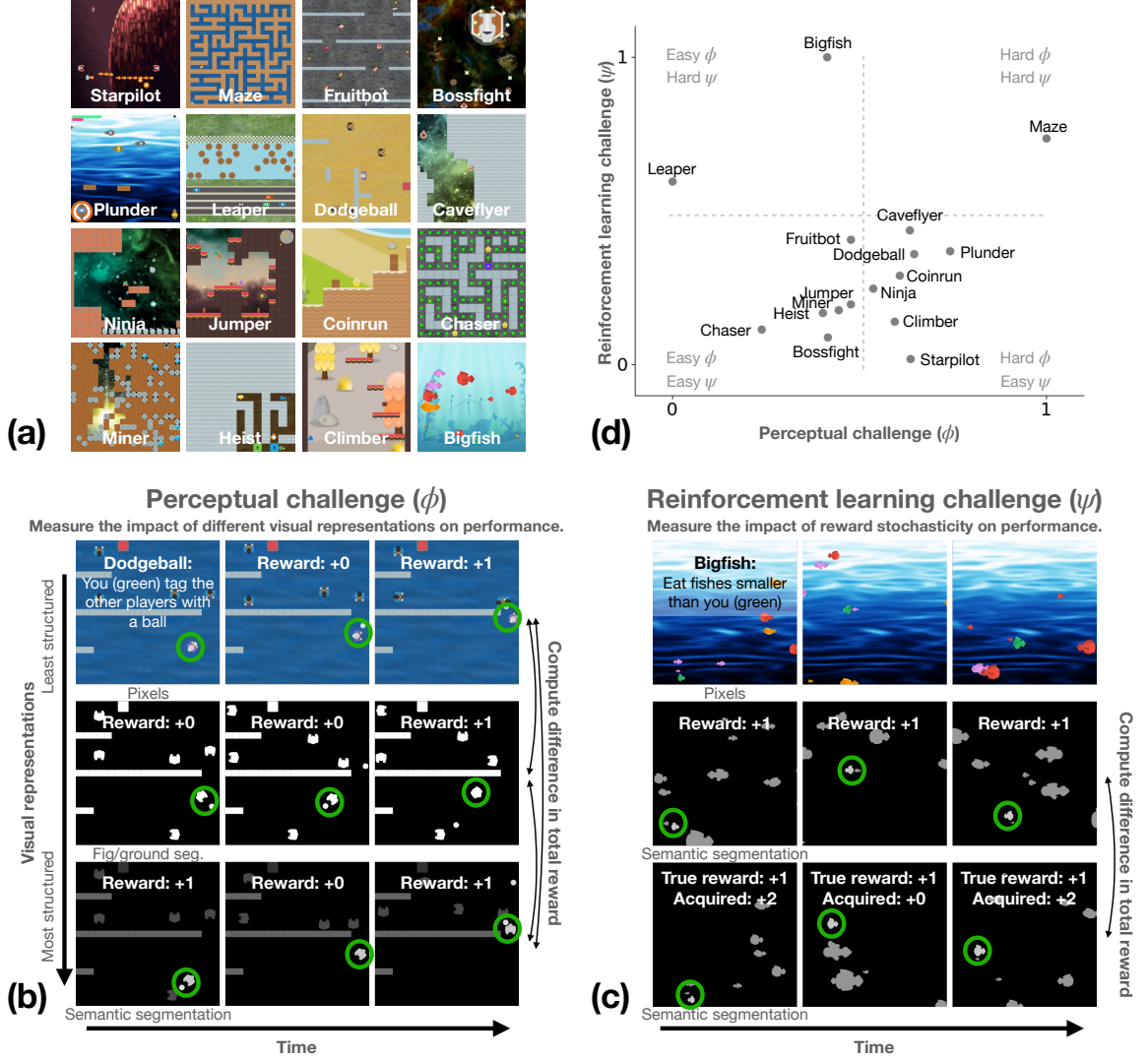


Figure 4.3: **The Learning Challenge Diagnosticator (LCD)** separately measures the reinforcement learning or the perceptual challenge of an environment and task. (a) We use LCD to compute perceptual and reinforcement learning challenges in each game of the *Procgen* Benchmark. (b) Perceptual challenges are assessed by comparing the total reward accrued by agents that learn over visual representations with different amounts of structure: pixels, figure/ground segmentation masks, and semantic segmentation masks (see Appendix. 4.4.7). (c) Reinforcement learning challenges are computed by feeding agents perceptually organized scenes (i.e., semantically segmented; second and third rows), and then comparing the total reward accrued by one agent playing in an environment with stochastic manipulations of reward (randomly masked and modulated; third row) to another agent playing in the normal environment (second row). (d) The taxonomy of *Procgen* reveals each game’s relative perceptual and reinforcement learning challenge.

Contributions. The ability to systematically, autonomously, and reliably identify an appropriate strategy for learning a task in a given environment would enable the design of generalist dRL agents that can flexibly adapt to challenges as humans do. To make progress towards this goal, we introduce

the Learning Challenge Diagnosticator (LCD), a novel tool that identifies the specific computational challenges of an environment and task. When applied to video games, like those in *Procgen*, the LCD measures the difficulty of encoding task-relevant perceptual properties versus the difficulty of optimal policy discovery (i.e., determining which actions lead to rewarding outcomes).

- We develop a modified, parameterizable version of *Procgen*, to systematically manipulate the perceptual and reinforcement learning challenges presented to agents in each individual game. We provide our version of *Procgen* and all experimental code at <https://anonymous.4open.science/r/lcd-procgen/>.
- The LCD reveals a novel taxonomy of computational challenges in the games of our modified *Procgen*. Some are more visually complex, some are more challenging from the point of view of reinforcement learning, and others strain agents across both of these axes. This taxonomy is preserved across different dRL algorithms and the heterogeneity of the challenges associated with the *Procgen* games suggests that adopting a single “one size fits all” approach to learning, as is standard in dRL, is suboptimal.
- To address the visual challenges of *Procgen* games, we adopt a self-supervised visual front-end, which learns perceptual groups from motion cues in games without reward feedback. To test if reinforcement learning challenges can, in principle, be alleviated, we develop a proof-of-concept approach with reward shaping [42].
- We exploit the computational taxonomy of *Procgen* revealed by the LCD to shape the design of agents for each game. These agents learn more efficiently and perform significantly better than one-size-fits-all dRL agents, suggesting the potential for “adaptive” agents.

4.4.2 Related work

Representation learning in dRL Perhaps the main contribution of DQNs was to demonstrate that it is possible to jointly learn visual representations and policies through reward maximization from a *tabula rasa* to achieve high scores on Atari games [20]. Despite this extraordinary achievement, the extrinsic rewards available to algorithms like DQNs have been found to yield poor sample complexity, limit model scale, and cap maximum performance [43, 44]. For this reason, there has been a growing number of accounts showing that standard reward learning can be augmented with auxiliary learning objectives and/or pre-training via self-supervision to take steps towards ameliorating limitations of reward-based learning [43–48]. Relatedly, it has been shown that initializing agents with structured visual representations can lead to faster learning and better performance in reinforcement learning [44, 46, 49, 50].

Measuring visual challenges in artificial intelligence An essential contribution that the cognitive sciences have made to artificial intelligence over the past decade has been the introduction of computational challenges that are easy for humans to solve but strain the capabilities of neural

networks. These challenges have identified deficiencies in the solutions deep neural networks (DNNs) tend to learn for contour tracing [51–53], segmentation [52], object tracking [54], object recognition [55, 56], and visual reasoning problems [40, 41, 57]. The essential lesson from these studies is that measuring model performance on data sampled from the same distribution that training data came from is not sufficient for establishing challenges and measuring progress on them [58]. Instead, it is critical to test models on out-of-distribution samples to find and evaluate challenges [54–56, 59]. We adopt this strategy to measure dRL agent performance in this work.

Biologically inspired mechanisms for improving neural networks There are multiple examples of the DNN challenges being resolved by the introduction of biologically inspired mechanisms. Mechanisms for attention have made significant contributions by enabling DNNs to limit noise and clutter, amplify task-relevant features, and subsequently achieve better performance and sample efficiency during learning [60]. Others have found that horizontal connections, inspired by those in the early visual cortex of primates, can resolve DNN limitations in solving contour tracing tasks [61]. There is also evidence that forcing dRL agents to acquire compositional abstractions, inspired by those described in the cognitive sciences, helps them learn policies that generalize to novel environments [30]. The mechanisms we propose to solve the computational challenges revealed by the LCD in *Progen* are similarly inspired by solutions to those problems adopted by primates and humans.

Metrics for measuring progress in dRL Progress in dRL has been classically measured by the aggregate rewards and sample efficiency of models in games [62–64]. The *Progen* benchmark took a major step forward beyond what is standard in the field by measuring agent performance on game parameterizations that fell outside of the distribution used for training. While current metrics can adjudicate between different algorithms based on performance in a game, they do not offer insights into the specific computational challenges that agents face. Our LCD therefore represents a major conceptual leap beyond current approaches to measuring progress in dRL, by quantifying the specific perceptual and reinforcement learning challenges found in a given game, and enabling dRL researchers to more precisely resolve the deficiencies of agents.

4.4.3 The Learning Challenge Diagnosticator

The ability to identify and measure the computational challenges a dRL agent faces is an important step toward designing better algorithms for reinforcement learning. However, to the best of our knowledge, this problem has not yet been addressed in the field. The LCD achieves this goal through a three-step procedure, which separately identifies the perceptual challenge and reinforcement learning challenge offered by any specific game: (i) We begin by modifying a game to put its perceptual representations and reinforcement learning problem under experimental control. (ii) Next, a set of agents learn to play different versions of a game. In each version, either the perceptual challenge is perturbed while the reinforcement learning challenge is held constant, or vice versa. These perturbations are implemented by sampling either a different perceptual representation of the game or reward scheme. (iii) Finally, we measure the integrated reward trajectories of agents in

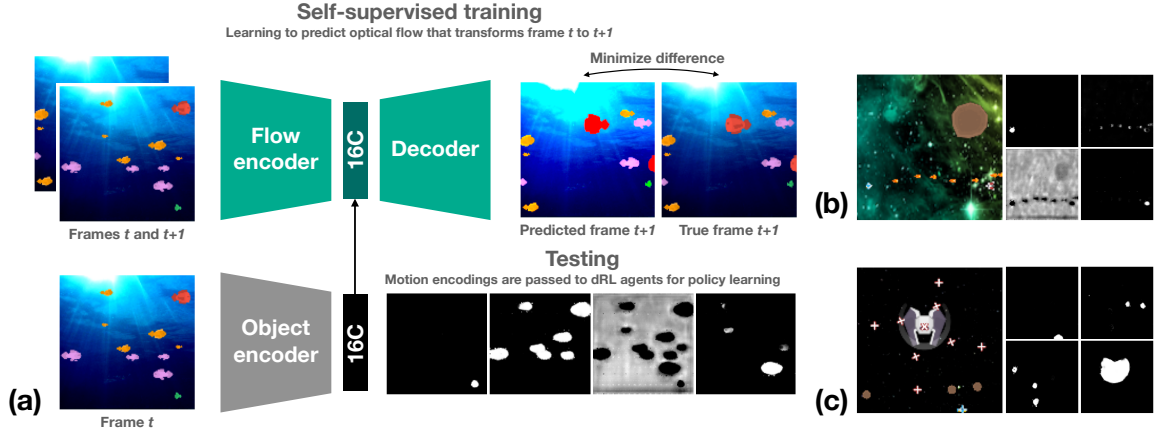


Figure 4.4: **A DNN that is self-supervised to predict future frames also learns a concomitant representation of object-ness that is useful for reinforcement learning.** (a) Our routine for self-supervising perceptual representations that can solve challenges identified by the LCD utilizes three networks: a ResNet18 [65] for predicting the optic flow between frame t to frame $t + 1$ (*Flow encoder*), a ResNet18 feature pyramid network (FPN) with a 16-channel recurrent neural network [61] to encode the content of frame t (*Object encoder*), and an FPN-ResNet18 decoder that receives the two encoder’s outputs to transform frame t to frame $t + 1$ via a differentiable warping module. After optimizing for next-frame prediction, the *Object encoder* learns object-like representations that can be used for reinforcement learning without additional training. This approach generates reasonable segmentations of the important objects in *Procgen* games, including Bigfish, Starpilot (b), and Bossfight (c).

response to each type of perturbation. By separately averaging the scores of agents facing perceptual perturbations and reward perturbations, we compute scores describing the perceptual challenge (ϕ) and the reinforcement learning challenge (ψ) of each game (Fig. 4.3b and c).

Procgen benchmark. We applied the LCD to *Procgen* [39], a challenging dRL benchmark consisting of 16 games, each with distinct graphics and gameplay. *Procgen* is open-source and implemented in Python and C++, which made it possible to customize a variety of game parameters for implementing the LCD. The parameters we manipulated were game assets (sprites and backgrounds) and reward functions.

Procgen generates game levels procedurally, enabling precise control over the levels used for training versus testing and as a result, proper tests of generalization. We followed the training and testing protocol outlined in [39] in each of our experiments. In brief, all agents were trained for 200M steps across 500 training levels and evaluated on *Procgen*’s held-out games to measure generalization.

dRL algorithms. We apply the LCD to agents trained with either the standard and popular proximal policy optimization [66] (PPO) or phasic policy gradient (PPG) [67], a more recent algorithm that attained state-of-the-art performance in *Procgen*. PPO hyperparameters were selected to match those used in [39] for *Procgen*’s hard mode. Similarly, for PPG we used the hyperparameters from [67]. The walltime for a single training run of one of these agents averaged 24 hours per game on a standard

16-core Intel Xeon Gold 6242 CPU with a 24G Titan RTX GPU. We ran ~ 450 experiments across 32 GPUs, which took over 10K GPU hours of computing time. Agent performance at test time was computed using the interquartile mean (IQM) of the final returns [63].

LCD: Perceptual challenge. To perturb the perceptual complexity in a game and measure its perceptual challenge, we varied the visual input provided to an agent when learning to play the game. We tested the impact of learning on three different types of visual inputs, which varied in the amount of structure in the environment they conveyed to agents: the original frame, figure-ground segmentations of each frame, or semantic segmentations of each frame (Fig. 4.3b,c). Figure-ground segmentations labeled background and object pixels differently, whereas semantic segmentations labeled each element of the frame differently, with consistent labels across frames (see Appendix. 4.4.7).

We measured the perceptual challenge of each game by comparing the performance of three agents trained to solve it, where each agent learned a policy over one of the three different types of visual inputs (Appendix Fig. 4.8). In this condition, rewards and the reinforcement learning challenge in games were kept as normal. We began by computing performance for each agent as the area under the cumulative reward curves (AUC) achieved by agents during generalization (Appendix Fig. 4.10), which implicitly captured both maximum performance and the sample efficiency needed to get there. Next, we computed the relative change in AUC from the original frame to figure-ground and semantic AUCs, and took the average of these ratios as our final score (Fig. 4.3b). The perceptual challenge score of each game was then normalized to the range $[0, 1]$ according to the min and max across all games (Appendix 4.4.7). A game with a score approaching 1 indicates that it presents a greater perceptual challenge for reward-based learning, whereas a small score means that reward-based discovery of task-relevant features is sufficient to perform well on that task.

LCD: Reinforcement learning challenge. To perturb reinforcement learning in a game, we manipulated the sparsity of rewards available to an agent. We did this by decreasing the probability p of receiving a reward at every rewarding event, while also scaling reward magnitudes by $1/p$ to preserve the expected magnitude of rewards, and by extension the optimal policy, across all perturbations (Appendix 4.4.7). We used reward perturbations of $p \in \{1, 0.75, 0.50, 0.25\}$ (Appendix Fig. 4.9). All agents were given semantic segmentations as inputs, which minimized the perceptual challenge of games and controlled for potential perceptual confounds, such as how the perceptual complexity of the pixel inputs of a given game may non-linearly interact with this reinforcement learning perturbation.

We measured the reinforcement learning challenge of each game by comparing the performance of agents trained to solve each of the reward perturbations (Appendix Fig. 4.11). First, we again computed AUCs of the generalization reward curves for each agent, as above, but here for different values of p . Next, we computed the absolute AUC change from an agent trained on one p to the next and averaged across all changes, yielding the score (Appendix 4.4.7), which was normalized to $[0, 1]$ in the same way as . A score close to 1 means that it presents a significant reinforcement learning challenge. Conversely, a game with a small score indicates that reward sparsity is not the primary challenge.

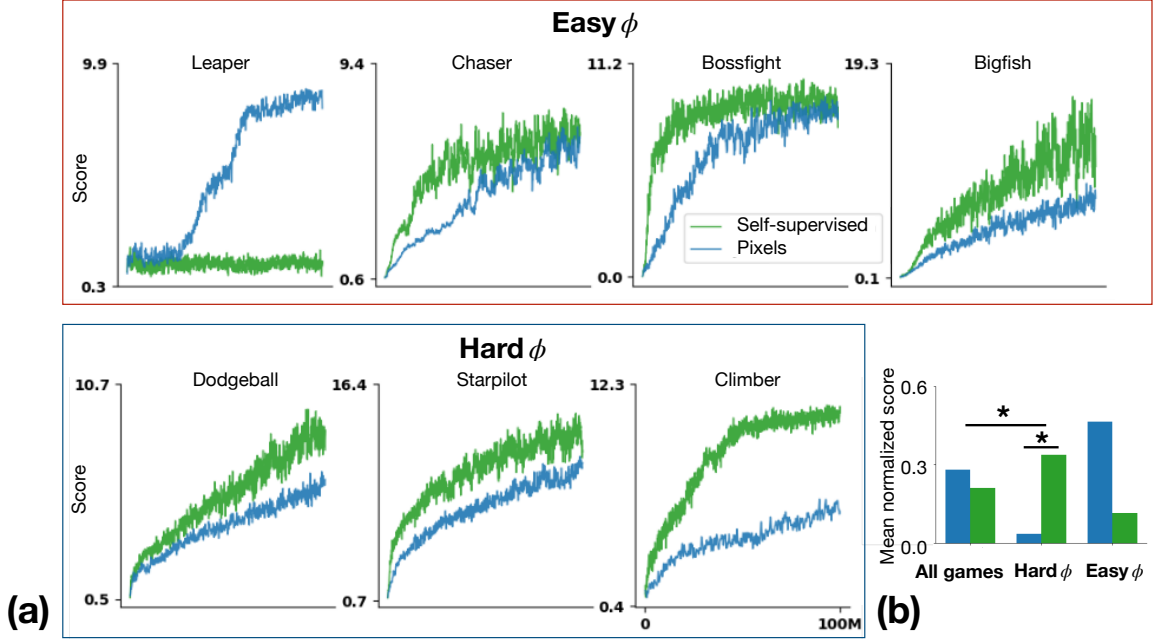


Figure 4.5: **Self-supervised pre-training for learning object-ness addresses perceptual challenges in *Progen* identified by the LCD.** (a) We measured the impact of our self-supervised perceptual front-end for learning games classified as having perceptual challenges (Hard ϕ) or not (Easy ϕ). The game which is helped the least by the front-end, “Leaper”, requires agents to recognize colors – a feature that is not preserved in the front-end. (b) When looking at aggregate performance across the entirety of *Progen*, the benefit of pre-training is only seen on those games that the LCD deems as perceptually challenging (Hard ϕ), indicating that the current state of progress in dRL is being slowed by an inability to find benchmarks (or subsets of benchmarks) which are aligned with the algorithmic solutions being tested. One-tailed t -tests assessing the effect of learning from the perceptual front-end versus pixels are denoted by lines, $* = p < 0.05$.

Diagnosing the computational challenges of the Progen benchmark. We began by applying the LCD to PPO agents trained to solve each game in *Progen*. Doing so revealed a clear taxonomy of the computational challenges of each game (Fig. 4.3d): Fruitbot, Jumper, Miner, Heist, Chaser, and Bossfight are, in relative terms, perceptually easy and have easy reinforcement learning. Caveflyer, Dodgeball, Plunder, Coinrun, Ninja, Climber, and Starpilot are perceptually hard and have easy reinforcement learning. Bigfish and Leaper are perceptually easy and have hard reinforcement learning, and Maze is challenging for perception and reinforcement learning. The challenges of Maze, in particular, are likely driven by the need to trace and plan paths through a maze; routines which are known to be challenging for neural networks and draw upon feedback mechanisms in humans [61, 68–70].

4.4.4 Validating LCD taxonomies by building better agents

Our general approach for validating the computational taxonomy of the LCD is to demonstrate that the gains of targeted solutions to the challenges coincide with the LCD’s predictions. We develop

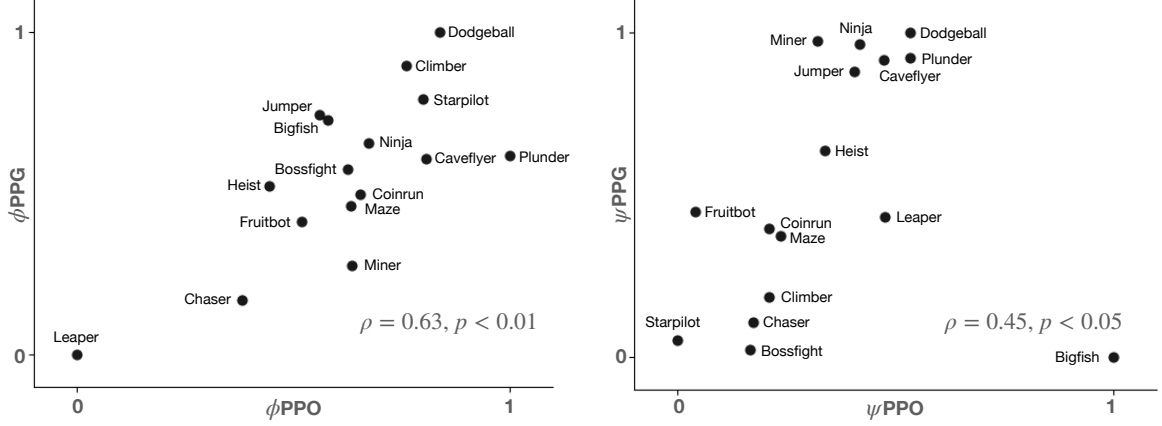


Figure 4.6: **LCD measurements of the perceptual and reinforcement learning challenges in *Procgen* translate across dRL algorithms.** Measurements of visual challenges ϕ (left) and reinforcement learning challenges ψ (right) faced by PPO agents were significantly correlated with the challenges of PPG agents, indicating that results derived from applying the LCD to simpler and lower-overhead algorithms can guide the development of agents pushing the state-of-the-art.

separate solutions for perceptual challenges and reinforcement learning challenges and describe those methods and results here.

Developing solutions to perceptual challenges. We began by constructing a new perceptual front-end that we believed could address the perceptual challenges of *Procgen* games. Inspired by the reliance of humans and other animals on optic flow to effectively and efficiently navigate through their environments [19, 71], we turned to motion perception as the base inductive bias for our agents. Specifically, we rely on the motion of objects in the world to extract perceptual groups that could instruct appropriate behavior.

It has been found that training DNNs to predict the optic flow between successive frames of video can induce the ability to segment object-like superpixels from complex scenes [72]. Here, we build off this prior work to develop a self-supervised pre-training strategy for solving perceptual challenges in *Procgen*. Our architecture consists of three separate DNNs: (i) a ResNet18 [65] feature pyramid network (FPN) [73] for predicting the optic flow (*Flow encoder*) between two frames from successive timepoints of a sequence, frame t and frame $t + 1$, (ii) a biologically inspired recurrent neural network [61] (*Object encoder*) for encoding the contents of frame t , and (iii) a fully convolutional [74] decoder that learns how to use the two encoders’ outputs to match frame t to frame $t + 1$ via a differentiable warping module.

This model was pre-trained to learn visual representations in a purely self-supervised manner, by minimizing the mean squared error between two successive frames sampled from the seven games of *Procgen* which had consistent motion (Chaser, Leaper, Dodgeball, Climber, Bossfight, Starpilot, and Bigfish). We sampled frames for training from 14,000 videos (2,000 per game) in batches of 4 for 50k iterations using the Adam optimizer [75] and a learning rate of $1e - 4$. As an

alternative to our approach for self-supervision, we also evaluated the effectiveness of state-of-the-art features for one-shot recognition from a clip-ViT-B-32 model pre-trained on 400M natural images and captions [76].

Perceptual challenge evaluations. We evaluated the LCD predictions of the perceptual challenges ϕ in each game by testing if preprocessing frames with our self-supervised front-end improved agent performance more for games with higher values of ϕ (*i.e.*, the front-end was fixed and not trained with reward). Because our self-supervised front-end relied on the game motion to extract perceptual groups, we focused our analysis on the same seven games with dynamic elements that it was pre-trained on. As predicted by the LCD, our front-end improved performance significantly more for perceptually challenging games (hard ϕ) than perceptually simple games (easy ϕ ; Fig. 4.5b). Overall, the benefit of this front-end correlated significantly with the ϕ predicted for each game (Spearman’s $\rho = 0.79$ between ϕ and the AUC of performance curves, $p < .05$).

To understand the extent that our results merely demonstrate the value of pre-trained visual representations, we repeated our analysis while using a pre-trained CLIP as the perceptual front-end for agents. But while CLIP embeddings yield state-of-the-art performance in one-shot classification [76], they did not help agents learn more effective policies, faster (Appendix Fig. 4.14). In other words, LCD’s perceptual challenges are more readily addressed by algorithmic solutions that induce object-like representations.

Guiding algorithmic development with the LCD. Given the success of our self-supervised perceptual front-end, an obvious approach towards building better agents is to simply use this front-end on every game. However, the impact of our front-end on performance was significantly blunted when measured on all games (the difference between green bars in Hard ϕ versus All Games, Fig. 4.5b). This finding points to a larger problem in algorithmic development for RL: without a reliable approach for diagnosing the specific computational challenges in a game, the effectiveness of reasonable solutions can be blunted or even appear to hurt agents, as our front end nominally does here.

Reward shaping for resolving reinforcement learning challenges. The reinforcement learning challenges revealed by the LCD are due to the sparsity of rewards available to agents. As a partial solution to this problem, we investigated the impact of “reward shaping” [42] on performance. Reward shaping is a technique from animal training where supplemental rewards are provided to make the learning problem easier [77]. We adopted this for three games amenable to reward shaping: Heist, Leaper, and Maze; the latter two LCD identified as having significant reinforcement learning challenges.

In Heist, agents must collect three keys and unlock their matching doors before they can reach the rewarding gem. We provided agents with additional rewards when they collected a key or opened a door. In Leaper, agents must safely get across several lanes of roads followed by several rows of

rivers. Lanes have moving cars agents must avoid, while rivers have moving logs agents must stay on. We gave agents additional rewards whenever they reached a new lane or river.

In Maze, agents must wander through a complex maze to get to a reward of cheese; however, the topological complexity of the mazes can deem rewards difficult to discover. To address this problem, we encourage exploration of novel locations through intermediate rewards [78, 79]. In all of these cases, agents did not get these auxiliary rewards during test time.

Reinforcement learning challenge evaluations. Agents trained with reward shaping on Heist ($t = 1.69, p < .05$), Leaper ($t = 41.03, p < .001$), and Maze ($t = 2.69, p < .01$) performed significantly better than those trained on the normal versions of each game (all tests are one-tailed t -tests; Appendix Fig. 4.15). Moreover, the improvements in performance from reward shaping significantly correlated with the ψ values of each game tested here (Spearman’s $\rho = 0.92, p < .001$).

The LCD-derived taxonomy translates across dRL algorithms. Our findings thus far indicate that an LCD calibrated with PPO agents makes reliable predictions about the challenges of individual games in *Procgen*. But despite the popularity of PPO, it has been surpassed by more recent dRL algorithms, like PPG, which achieved state-of-the-art performance on *Procgen*. To what extent does the LCD taxonomy derived from PPO translate to more effective dRL algorithms? We tested this question by applying the LCD to agents trained with PPG (Appendix Figs. 4.12, 4.13) and found that its predictions were significantly correlated with those from an LCD calibrated on PPO agents (Fig. 4.6). This means that our PPO-derived taxonomy can be relied on prospectively for algorithmic development on *Procgen*.

The whole is only sometimes greater than the sum of the parts in dRL. Training perception-for-action, end-to-end, is the standard approach to dRL ever since the introduction of DQNs. Here, we tested whether agents trained on *Procgen* experienced a superlinear benefit of co-training perception and action or not. We did this by computing two performance differentials. In both cases, the *baseline* was a PPO agent trained on pixels. (i) the performance of a PPG agent trained on semantic segmentation inputs *minus* baseline, and (ii) the performance of a PPG agent trained on pixels *minus* baseline *plus* the performance of a PPO agent trained on semantic segmentation inputs *minus* baseline (Fig. 4.7).

Co-training perception for action helped performance in 6 of 16 games: Leaper, Maze, Chaser, Jumper, Starpilot, and Bigfish. Co-training did not help in 10 of 16 games: Heist, Dodgeball, Fruitbot, Miner, Coinrun, Bossfight, Plunder, Ninja, Caveflyer, and Climber. This finding poses an intriguing prospect. It suggests the existence of different “minibenchmarks” within *Procgen* that researchers should tap into, depending on whether they wish to quantify “improvements” to an agent’s policy learning algorithm, its representation learning system, or synergies between the two.

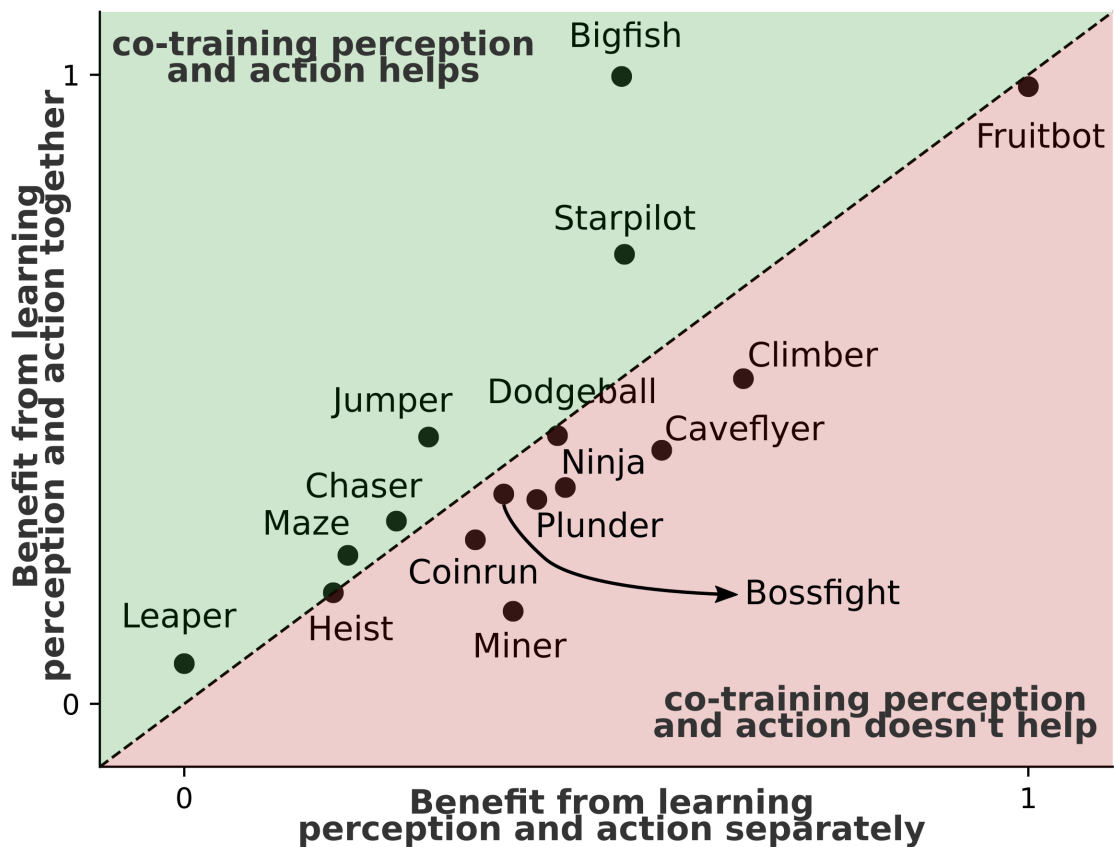


Figure 4.7: A subset of games in *Procgen* see a synergistic benefit from co-training perception and action/policies end-to-end. We compared the magnitude of improvement over baseline (PPO/pixels) agents experienced when they were given an ideal reinforcement learning algorithm (PPG) and perceptual representations (semantic segmentation) to the additive improvement of an agent given PPG over pixels and an agent given PPO and semantic segmentation inputs. Only 6/16 games benefit from co-training.

4.4.5 Discussion

Ever since the introduction of the DQN, the field of dRL has consistently succeeded in reaching and exceeding human performance on defined tasks: beating grand masters at the game of Go, or exceeding the average performance of humans on the Atari benchmark. However, even in these successes, it was clear that certain games presented different challenges than others. Take “Montezuma’s Revenge”, which the original DQN could not solve. Subsequent analysis of Montezuma’s Revenge led to the conclusion that the failures of the DQN were because of its long time horizon and sparse rewards; challenges which, these days, have positioned it as a critical test for dRL algorithms [80–82]. We believe that the field of dRL can benefit from a systematic approach to diagnosing such computational challenges in any benchmark. Our LCD tool is a major step towards this goal.

By applying our LCD to the *Procgen* benchmark, we discover a novel taxonomy of those games, organized according to the relative perceptual and reinforcement learning challenges of each. This taxonomy delivers two prescriptions for algorithmic development in dRL. First, because the computational challenges of games in benchmarks like *Procgen* are not i.i.d., there will be low signal for adjudicating between algorithms designed to improve a problem that is not over-represented in the benchmark. Second, although humans are capable of learning perception-for-action, and dRL agents generally learn in this way, not all games in benchmarks like *Procgen* benefit from doing so. In the short-term, these problems can be addressed by using the LCD to select games in a benchmark that are aligned with the algorithmic design goals of researchers. In the long-term, there is a need in the field for new benchmarks that can tap into a wider range of computational challenges with less bias than is found in *Procgen*. We believe that our LCD is an important tool for advancing the pace of progress in dRL, and we release our experimental code to support this goal.

4.4.6 Reproducibility statement

We have taken several steps to ensure the reproducibility of our work. We have made all code for our modified *Procgen* environment and the self-supervised model for perceptual grouping available at <https://anonymous.4open.science/r/lcd-procgen/>. For PPO, we have used the code available at <https://github.com/openai/train-procgen> along with the hyperparameters found in [39]. For PPG, we have used the code at <https://github.com/openai/phasic-policy-gradient> and the hyperparameters in [67]. Finally, we have reported the computing resources used and the number of experiments run under “dRL algorithms” of Section 4.4.3.

4.4.7 Appendix

Network architecture

Following [39], we used the convolutional actor-critic IMPALA architecture described in [83]. This is a deeper network than what has been used previously, consisting of 15 convolutional layers, 16 residual blocks, and 1.6M parameters. We refer the reader to [83] for details. For raw pixel, semantic segmentation, and figure-ground segmentation, the number of input channels was fixed to three. This

controls the model architecture to be identical regardless of input. For semantic or figure-ground masks, we replicated the masked input three times and fed that into each of the three convolutional input channels. Our self-supervised visual model outputs 16 channel perceptual masks (at the same spatial resolution of the inputs) which are concatenated with the RGB values and fed into the actor-critic network for policy learning.

Reward normalization

Given that each task involves varying reward magnitudes, episode rewards r were normalized to facilitate comparison across tasks. Normalized rewards r' were computed as $\frac{r-R_{min}}{R_{max}-R_{min}}$, where R_{max} is the theoretical maximum score attainable in an episode while R_{min} is the average score a random agent would attain.

Stochastic reward feedback

Here, we prove that the expected return under policy π does not change under our manipulation of reward probability. We proceed in two parts. First, we show that the expected reward received from a rewarding event remains unchanged. Then we show that the action-value function $Q^\pi(s, a)$ for policy π remains unchanged. By implication, this means the optimal policy does not change either.

Let $R(s, a)$ denote the original reward function at state s and action a . In our manipulation, rewards are generated stochastically with probability $p(s, a)$, but reward magnitudes are rescaled by $\frac{1}{p(s, a)}$.

Definition 4.4.1 (Modified reward function). The modified reward function $\tilde{R}(s, a)$ is given by

$$\tilde{R}(s, a) \triangleq \frac{R(s, a)}{p(s, a)} \eta(s, a),$$

where $\eta(s, a) \sim \text{Bern}[p(s, a)]$.

The expected value of $\tilde{R}(s, a)$ is $R(s, a)$.

Proof.

$$\begin{aligned} \mathbb{E}_\eta [\tilde{R}(s, a)] &\triangleq \mathbb{E}_\eta \left[\frac{R(s, a)}{p(s, a)} \eta(s, a) \right] \\ &= \frac{R(s, a)}{p(s, a)} \mathbb{E}_\eta [\eta(s, a)] \\ &= R(s, a), \end{aligned}$$

since $\mathbb{E}_\eta [\eta(s, a)] = p(s, a)$. □

Using this lemma, we now prove our main result. Let $Q^\pi(s, a)$ denote the action-value function induced by the original reward function $R(s, a)$ while following policy π , and let $\tilde{Q}^\pi(s, a)$ denote the action-value function induced by $\tilde{R}(s, a)$ while following the same policy.

Theorem 3. The action-value function $\tilde{Q}^\pi(s, a)$ is given by $Q^\pi(s, a)$.

Proof. As is common practice in RL, we shall discount future rewards by $\gamma \in [0, 1]$ on each future time step. Before beginning, we shall introduce some notation. We shall denote the current time step by t . We shall denote the state and action at any time τ by s_τ and a_τ . Finally, by an abuse of notation, we shall use $\mathbb{E}_\pi[\cdot]$ to denote expectations over state-action trajectories generated by both the policy π and the state-transition probability $p(s'|s, a)$, where s' is the successor state.

We now proceed to the proof.

$$\begin{aligned}
\tilde{Q}^\pi(s, a) &\triangleq \mathbb{E}_{\pi, \eta} \left[\sum_{k=0}^{\infty} \gamma^k \tilde{R}(s_{t+k+1}, a_{t+k+1}) \middle| s_t = s, a_t = a \right] \\
&= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k \mathbb{E}_\eta \left[\tilde{R}(\tilde{s}, \tilde{a}) \middle| \tilde{s} = s_{t+k+1}, \tilde{a} = a_{t+k+1} \right] \middle| s_t = s, a_t = a \right] \quad (\text{law of total expectation}) \\
&= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k+1}, a_{t+k+1}) \middle| s_t = s, a_t = a \right] \quad (\text{Lemma 4.4.7}) \\
&= Q^\pi(s, a)
\end{aligned}$$

□

Note that in our experimental manipulations, we used a single reward probability p that was independent of s and a .

Performance change with increasing credit assignment difficulty

To quantify the average change with increasing credit assignment difficulty, we first computed the reward curve AUC $AUC(p)$ for $p = 0.25, 0.5, 0.75, 1.0$. We next computed the absolute change in AUC between successive values of p ; that is, $|AUC(p = 1.0) - AUC(p = 0.75)|$, $|AUC(p = 0.75) - AUC(p = 0.5)|$, and $|AUC(p = 0.5) - AUC(p = 0.25)|$. Finally, we averaged over these measures to get our final measure of average change.

Supplementary Figures

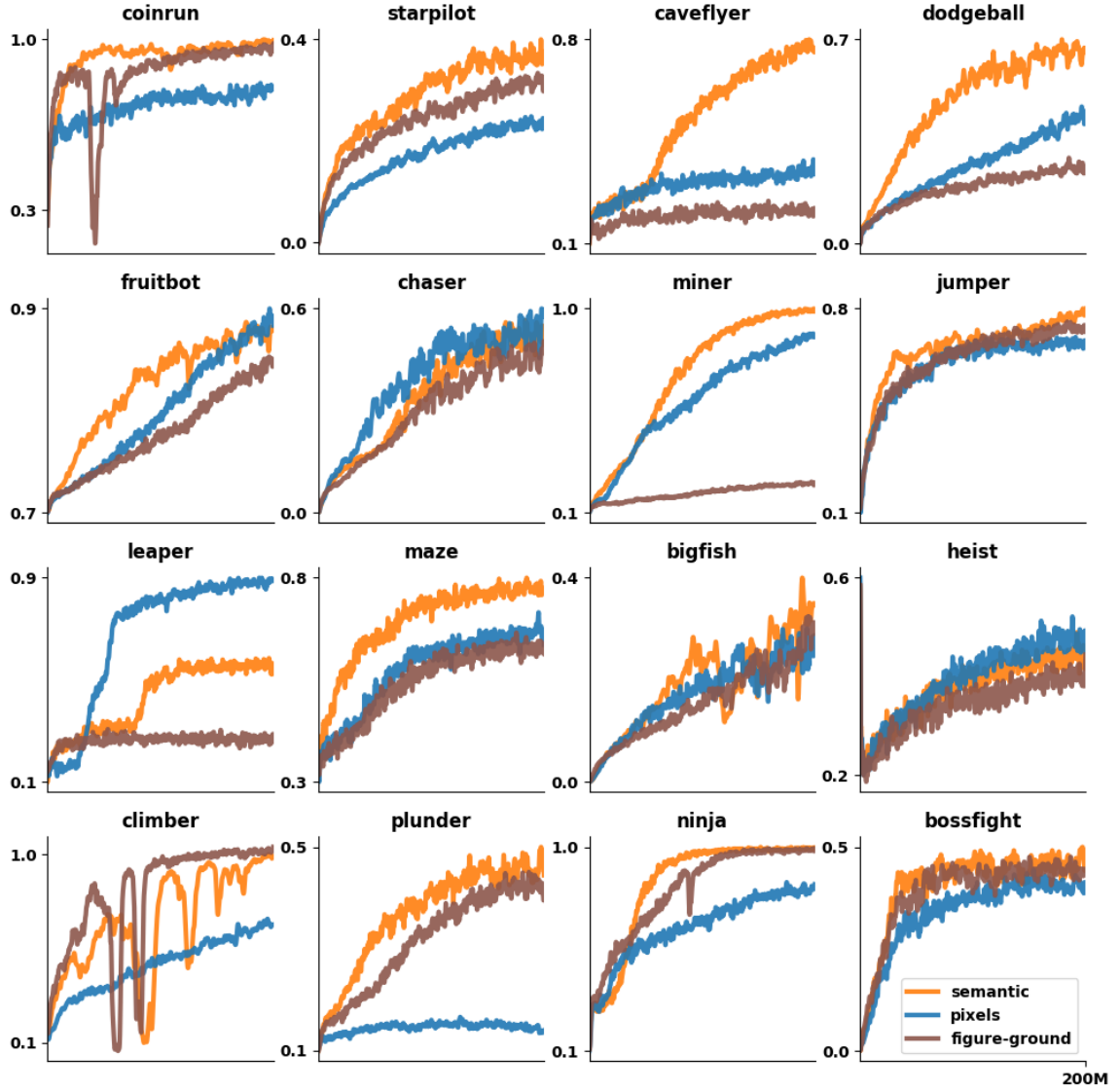


Figure 4.8: Normalized reward curves while systematically varying perceptual complexity. X-axis denotes the number of “interactions” an agent performs with its environment during training and Y-axis denotes normalized rewards. We trained all these agents for a total of 200M steps.

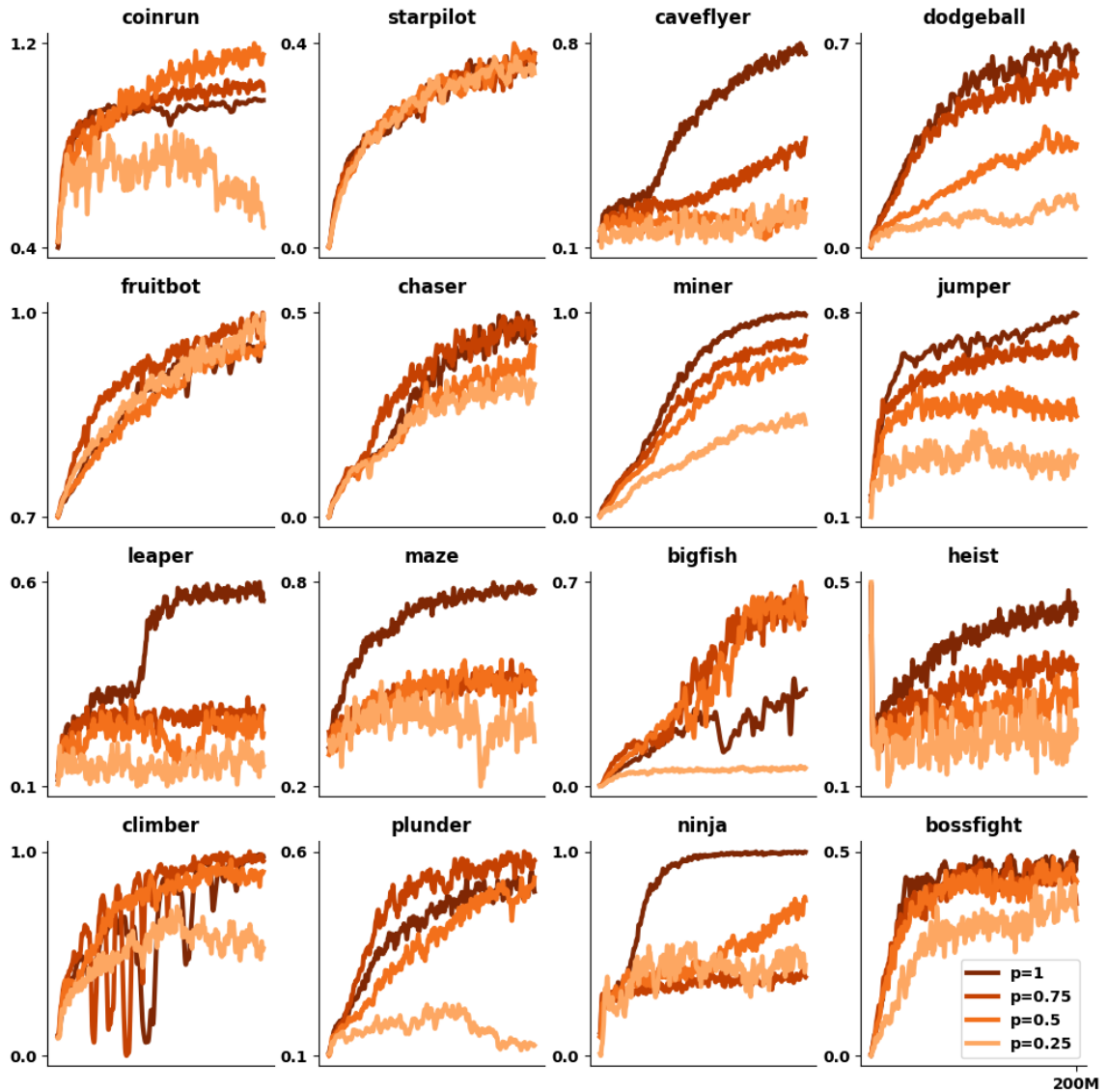


Figure 4.9: **Normalized reward curves while systematically varying reward stochasticity.** X-axis denotes the number of “interactions” an agent performs with its environment during training and Y-axis denotes normalized rewards. We trained all these agents for a total of 200M steps. The perceptual input to these agents were semantic representations. While performing evaluations, the reward mechanisms were set to the original configuration thus effectively only testing the impact of these perturbations on policy learning.

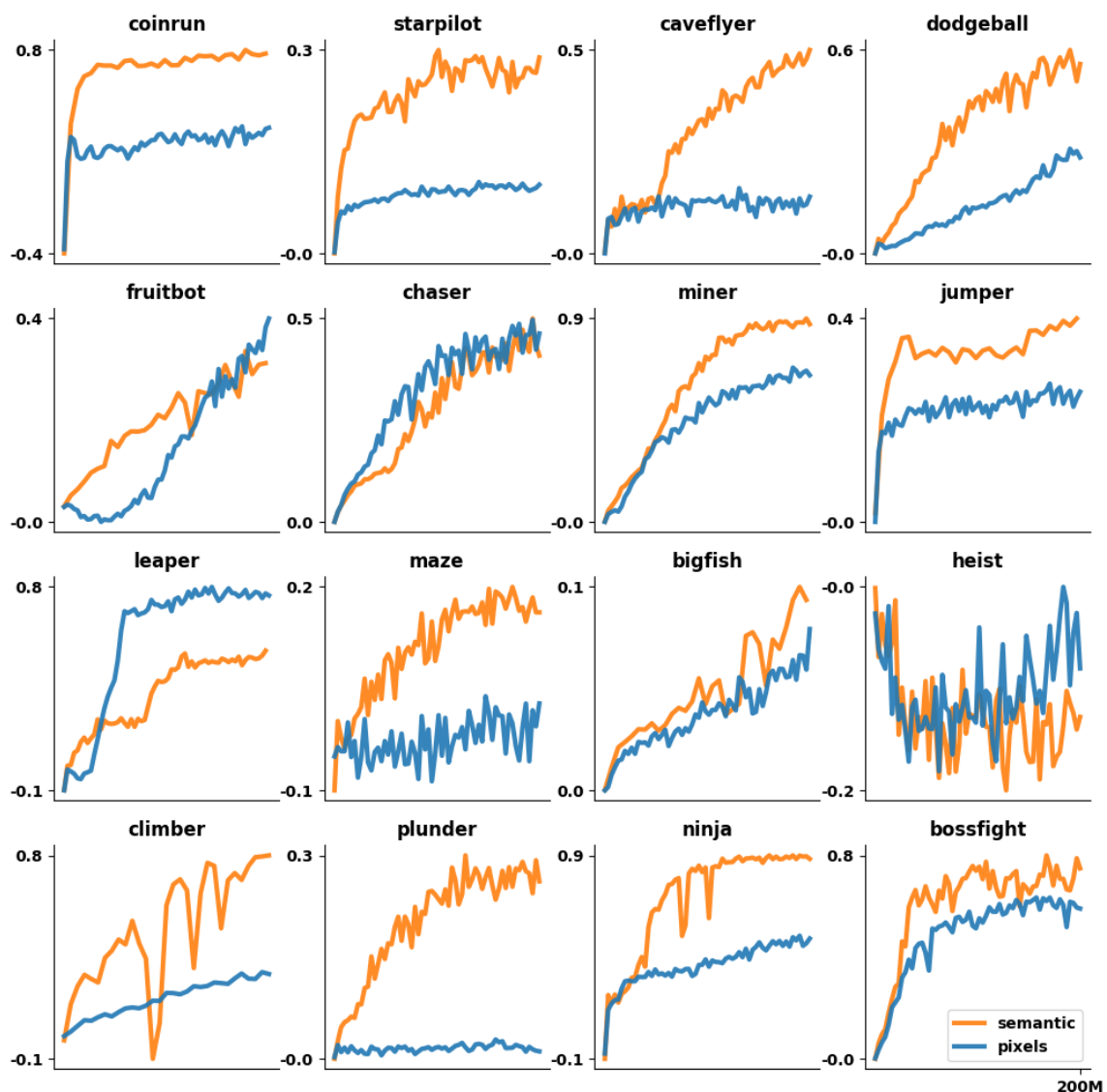


Figure 4.10: Normalized reward curves, on *i.i.d.* generalization levels, while systematically varying perceptual complexity. X-axis denotes the number of “interactions” an agent performs with its environment during training and Y-axis denotes normalized rewards. We trained all these agents for a total of 200M steps. Agents were evaluated approximately every 60K steps.

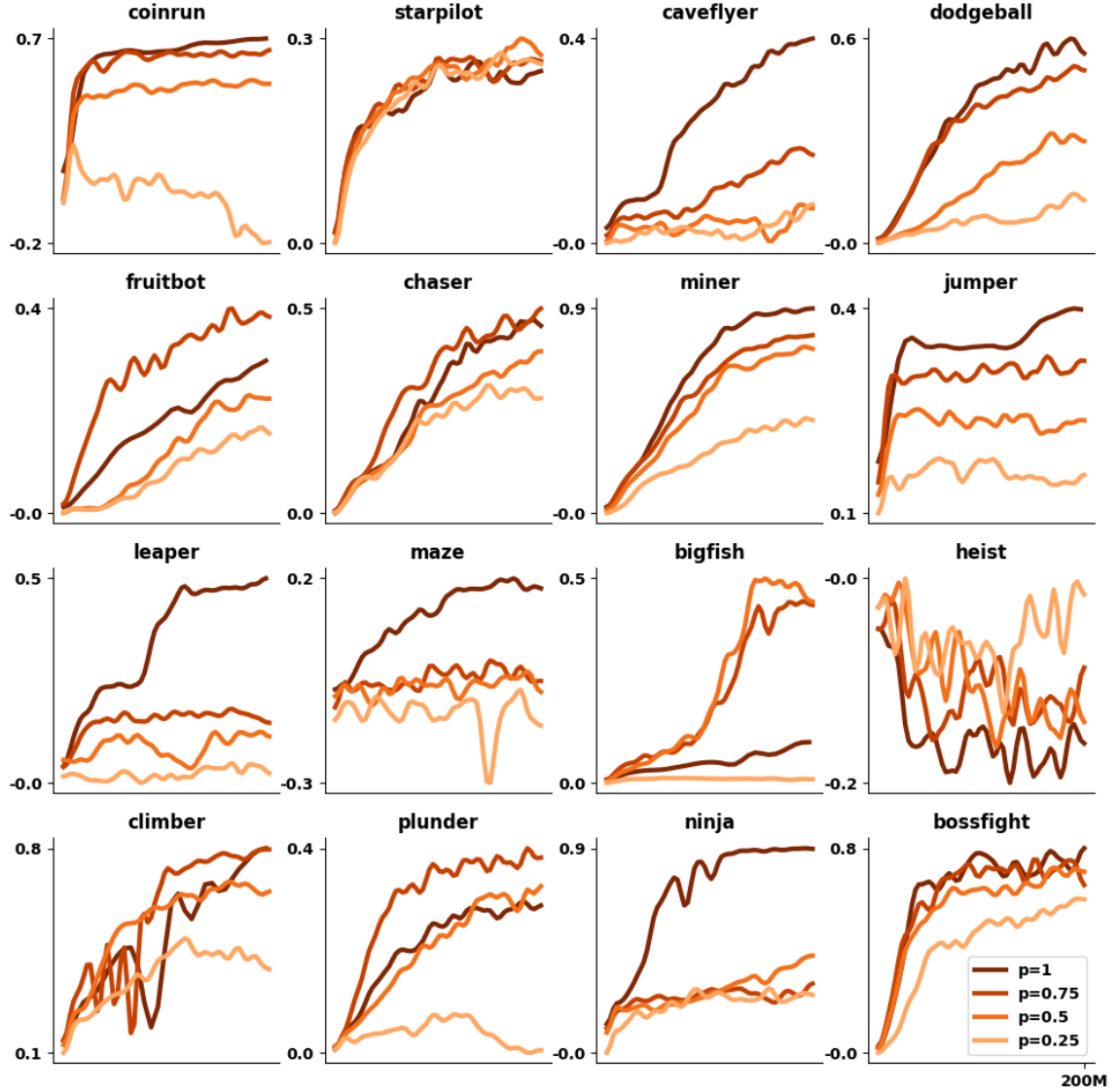


Figure 4.11: Normalized reward curves, on *i.i.d.* generalization levels, while systematically varying reward stochasticity. X-axis denotes the number of “interactions” an agent performs with its environment during training and Y-axis denotes normalized rewards. We trained all these agents for a total of 200M steps. The perceptual input to these agents were semantic representations. Agents were evaluated approximately every 60K steps. While performing evaluations, the reward mechanisms were set to the original configuration thus effectively only testing the impact of these perturbations on policy learning.

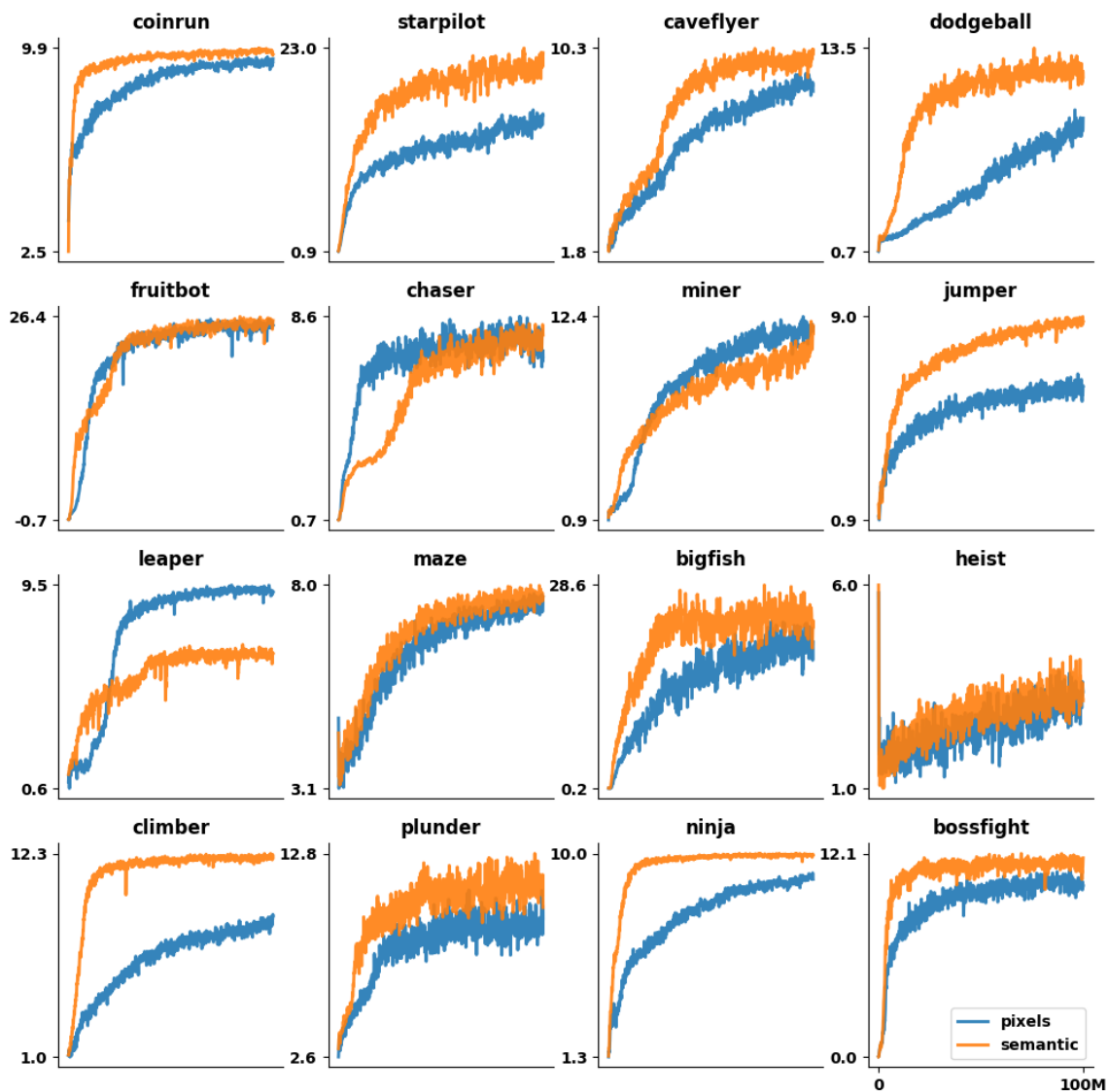


Figure 4.12: Training dRL agents with varying levels of perceptual complexity using the proximal policy gradient algorithm. X-axis denotes the number of “interactions” an agent performs with its environment during training and Y-axis denotes rewards on training levels. We trained these agents for a total of 100M steps.

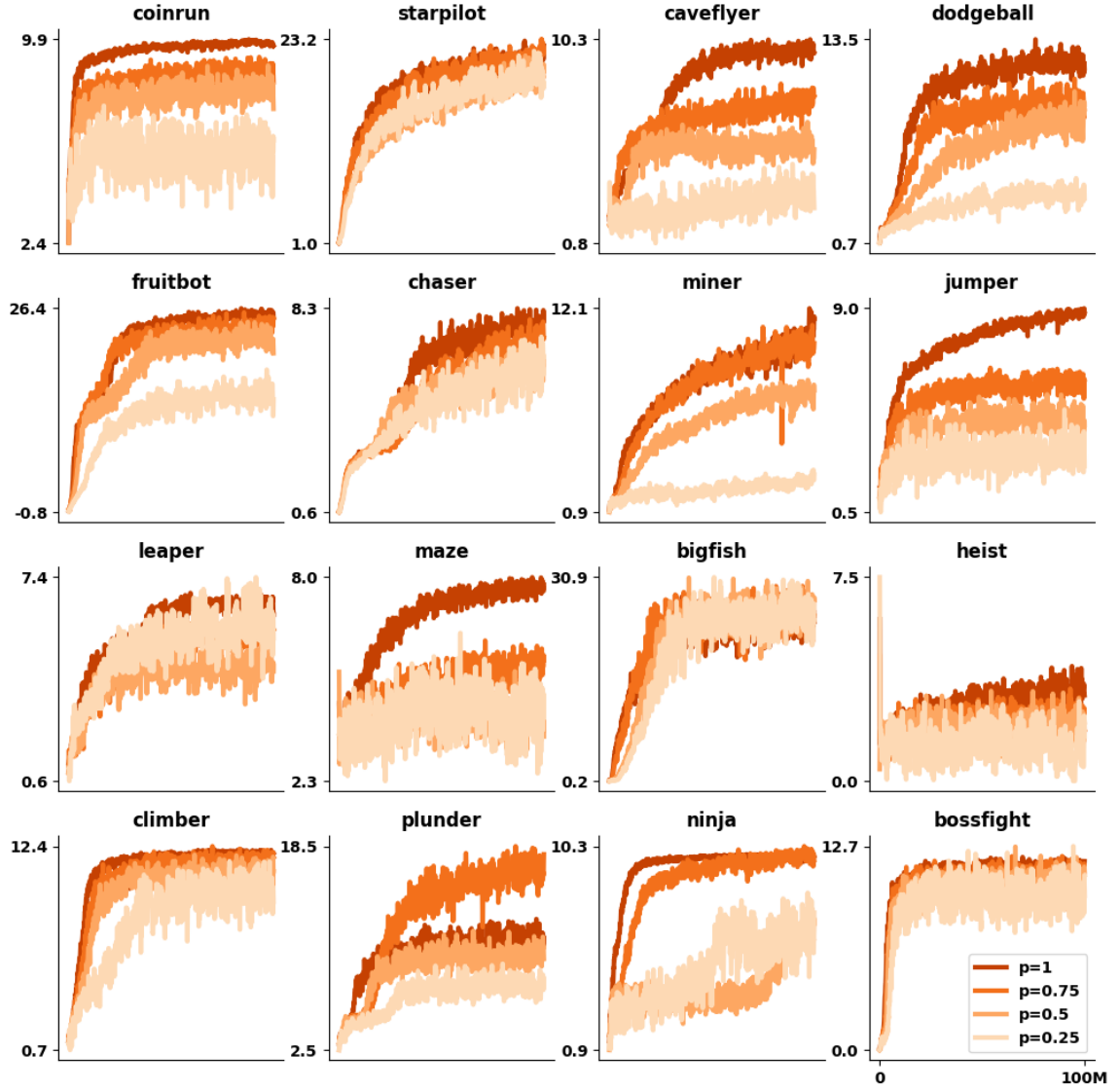


Figure 4.13: Training dRL agents with varying levels of stochastic feedback using the proximal policy gradient algorithm. X-axis denotes the number of “interactions” an agent performs with its environment during training and Y-axis denotes rewards on training levels. We trained these agents for a total of 100M steps. The perceptual inputs to these agents were semantic representations. While performing evaluations, the reward mechanisms were set to the original configuration thus effectively only testing the impact of these perturbations on policy learning.

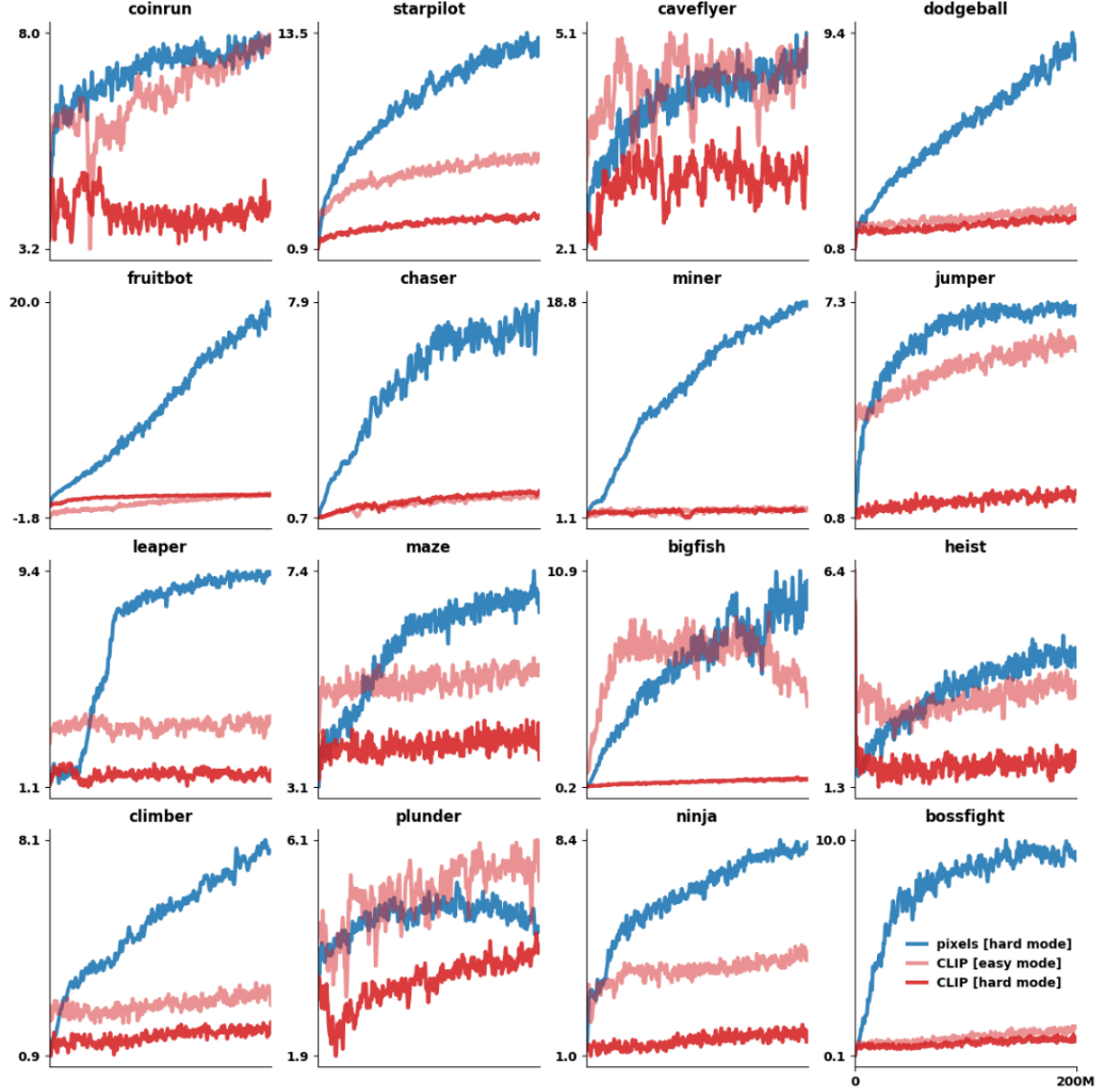


Figure 4.14: **Policy learning experiments in dRL agents operating on visual representations learned from natural language supervision.** We test the extent to which generalist visual representations support policy learning. We find that while CLIP [76], a transformer-based architecture, is state-of-the-art on zero-shot image classification, its representations do not support policy learning adeptly. We train agents on both the “easy” and “hard” versions of *Procgen*. While our agent was able to learn certain tasks in the “easy” mode, they predominantly struggled in the “hard” mode. X-axis denotes the number of “interactions” an agent performs with its environment during training and Y-axis denotes rewards. We trained these agents for a total of 200M steps.

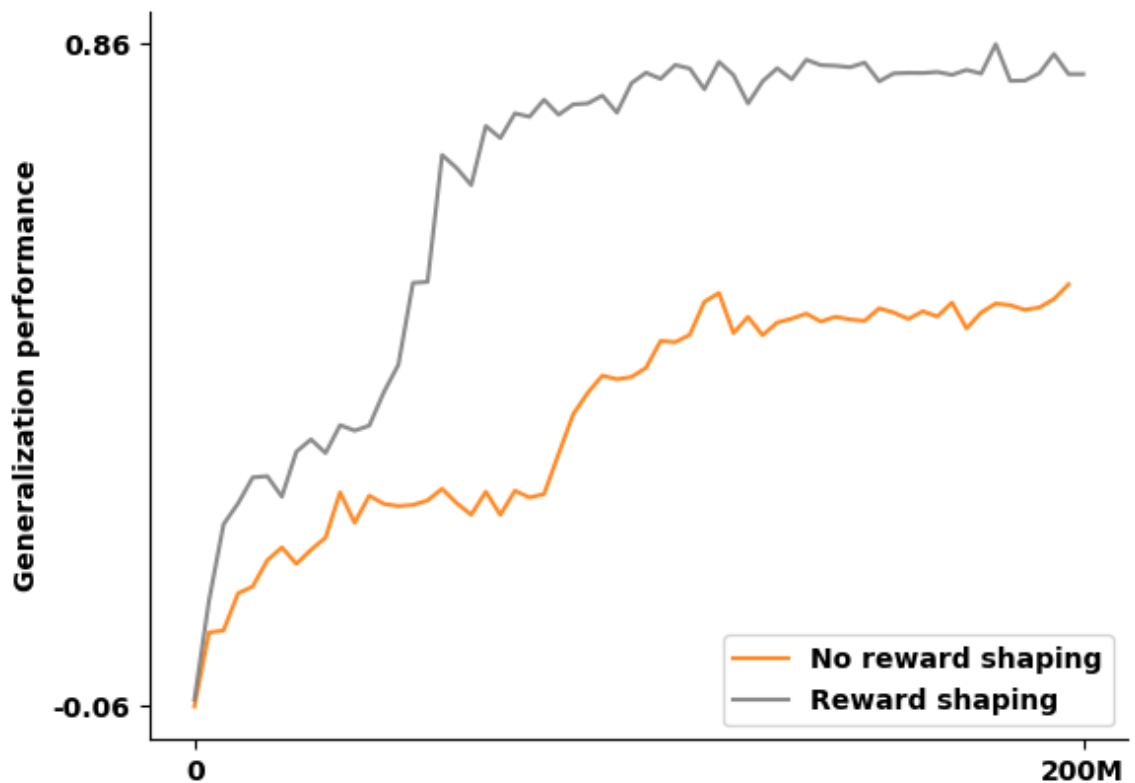


Figure 4.15: **The effect of “reward shaping” on Leaper – a high ψ task as predicted by our taxonomy.** In the traditionally challenging reinforcement learning problem we find that encouraging agents to discover sub-goals significantly improved both the sample efficiency and generalization capacity ($t = 41.03, p < .001$). X-axis denotes the number of “interactions” an agent performs with its environment during training and Y-axis denotes normalized rewards. We trained both these agents on semantic representations for a total of 200M steps. While performing evaluations, we revert to the original reward scheme to remain compatible for comparing to the “naive” model.

Specifics of our perceptual parameterizations of the *Procgen* benchmark

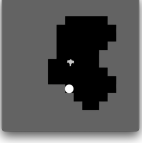
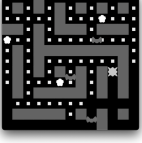
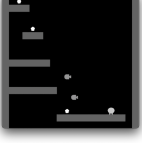
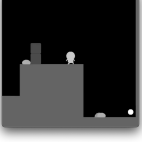
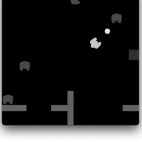
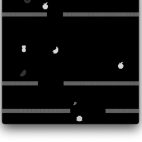
Image	Game	Mask Details
	bigfish	There are two entity IDs: (1) the player, and (2) all other fish.
	bossfight	There are seven entity IDs: (1) the player, (2) player projectiles, (3) the enemy, (4) enemy projectiles, (5) the enemy's shield, (6) meteors, and (7) explosions.
	caveflyer	There are eight entity IDs: (1) the player, (2) enemy ships, (3) player projectiles, (4) targets, (5) the goal landing pad, (6) meteors, (7) explosions, and (8) the surrounding terrain.
	chaser	There are seven entity IDs: (1) the player, (2) enemies, (3) enemies in their weakened state, (4) enemy eggs, (5) small collectible squares, (6) large collectible stars, and (7) the walls.
	climber	There are four entity IDs: (1) the player, (2) enemies, (3) the star rewards, and (4) the surrounding walls, ground, and platforms.
	coinrun	There are eight entity IDs: (1) the player, (2) enemies, (3) enemy barriers, (4) saw obstacles, (5) lava obstacles, (6) the goal star, (7) crates, and (8) the surrounding walls and ground.
	dodgeball	There are seven entity IDs: (1) the player, (2) player projectiles, (3) the enemies, (4) enemy projectiles, (5) the locked exit, (6) the unlocked exit, and (7) the walls.
	fruitbot	There are eight entity IDs: (1) the player, (2) player projectiles, (3) rewarding objects, (4) penalizing objects, (5) locks, (6) locked doors, (7) the goal, and (8) barriers and the terrain below and above the beginning and end of each episode, respectively.

Table 4.1: Outlining the “Semantic” organization of each environment in the *Procgen* benchmark. Part I

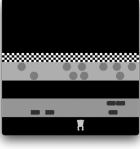
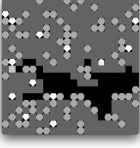
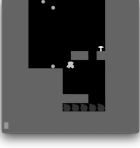
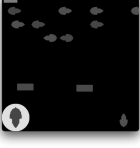
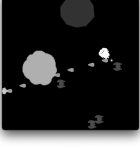
Image	Game	Mask Implementation
	heist	There are six entity IDs: (1) the player, (2-4) up to three unique keys and their corresponding locks, (5) the goal crystal, and (6) the walls.
	jumper	There are six entity IDs: (1) the player, (2) spikes, (3) the goal, (4) the needle of the compass and the bar indicating the distance to the goal, (5) the circle of the compass, and (6) the surrounding terrain.
	leaper	There are six entity IDs: (1) the player, (2) cars, (3) the road, (4) log platforms, (5) the water, and (6) the finish line.
	maze	There are three entity IDs: (1) the player, (2) the goal cheese, and (3) the walls.
	miner	There are five entity IDs: (1) the player, (2) boulders, (3) crystals, (4) the exit, and (5) dirt.
	ninja	There are seven entity IDs: (1) the player, (2) player projectiles, (3) the goal mushroom, (4) bombs, (5) explosions, (6) the bar indicating jump charge, and (7) the surrounding walls, ground, and platforms.
	plunder	There are eight entity IDs: (1) the player and ally ships, (2) player projectiles, (3) enemy ships, including the cue ship, (4) the circle behind the cue ship, (5) barriers, (6) explosions, (7) the time-countdown status bar, and (8) the points-accrued status bar.
	starpilot	There are eleven entity IDs: (1) the player, (2) player projectiles, (3) slow enemies flyers, (4) fast enemies flyers, (5) enemy flyer projectiles, (6) enemy turrets, (7) enemy turret projectiles, (8) meteors, (9) clouds, (10) the finish line, and (11) explosions.

Table 4.2: Outlining the “Semantic” organization of each environment in the *Procgen* benchmark. Part II

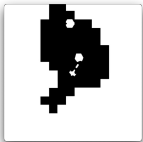
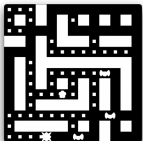
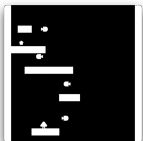
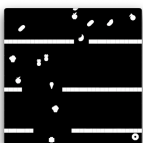
Image	Game	Mask Details
	bigfish	All entities share the same ID.
	bossfight	All entities share the same ID.
	caveflyer	All entities share the same ID.
	chaser	All entities share the same ID.
	climber	All entities share the same ID.
	coinrun	All entities share the same ID.
	dodgeball	All entities share the same ID.
	fruitbot	All entities share the same ID.

Table 4.3: Outlining the “Figure/Ground” organization of each environment in the *Procgen* benchmark. Part I

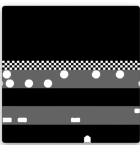
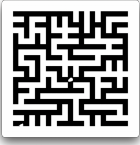
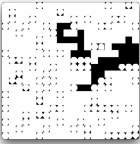
Image	Game	Mask Implementation
	heist	There are four entity IDs: (1-3) up to three unique keys and their corresponding locks, and (4) all other entities.
	jumper	There are three entity IDs: (1) the needle of the compass and the bar indicating the distance to the goal, (2) the circle of the compass, and (3) all other entities.
	leaper	There are two entity IDs: (1) the road and water, and (2) all other entities.
	maze	All entities share the same ID.
	miner	All entities share the same ID.
	ninja	There are two entity IDs: (1) the bar indicating jump charge, and (2) all other entities.
	plunder	There are five entity IDs: (1) the player and friendly ships, (2) enemy ships, including the cue ship, (3) the time-countdown status bar, (4) the points-accrued status bar, and (5) all other entities.
	starpilot	All entities share the same ID.

Table 4.4: Outlining the “Figure/Ground” organization of each environment in the *Progen* benchmark. Part II

4.5 References

- [1] Danique Jeurissen, Matthew W Self, and Pieter R Roelfsema. “Serial grouping of 2D-image regions with object-based attention in humans”. In: *Elife* 5 (2016), e14320.
- [2] Arezoo Pooresmaeili and Pieter R Roelfsema. “A growth-cone model for the spread of object-based attention during contour grouping”. In: *Current Biology* 24.24 (2014), pp. 2869–2877.
- [3] Pieter R Roelfsema and Roos Houtkamp. “Incremental grouping of image elements in vision”. In: *Attention, Perception, & Psychophysics* 73.8 (2011), pp. 2542–2572.
- [4] Pieter R Roelfsema et al. “Cortical algorithms for perceptual grouping”. In: *Annual review of neuroscience* 29.1 (2006), pp. 203–227.
- [5] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.
- [6] D P Kingma and J Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [7] Drew Linsley et al. “Stable and expressive recurrent vision models”. In: (May 2020). arXiv: 2005.11362 [cs.CV].
- [8] Alex Graves. “Adaptive computation time for recurrent neural networks”. In: *arXiv preprint arXiv:1603.08983* (2016).
- [9] D Knill D Kersten and A Yuille. “Introduction: A Bayesian formulation of visual perception”. In: *Perception as Bayesian inference* (1996), pp. 1–21.
- [10] Joshua B Tenenbaum et al. “How to grow a mind: Statistics, structure, and abstraction”. In: *science* 331.6022 (2011), pp. 1279–1285.
- [11] Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. “Simulation as an engine of physical scene understanding”. In: *Proceedings of the National Academy of Sciences* 110.45 (2013), pp. 18327–18332.
- [12] Tomer D Ullman et al. “Mind games: Game engines as an architecture for intuitive physics”. In: *Trends in cognitive sciences* 21.9 (2017), pp. 649–665.
- [13] Rebecca Keogh, Johanna Bergmann, and Joel Pearson. “Cortical excitability controls the strength of mental imagery”. In: *elife* 9 (2020), e50232.
- [14] Polina Iamshchinina et al. “Perceived and mentally rotated contents are differentially represented in cortical depth of V1”. In: *Communications biology* 4.1 (2021), pp. 1–8.
- [15] Aarit Ahuja, Theresa M Desrochers, and David L Sheinberg. “A role for visual areas in physics simulations”. In: *Cognitive Neuropsychology* (2022), pp. 1–15.
- [16] Diederik P Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: (Dec. 2014). arXiv: 1412.6980 [cs.LG].

- [17] Drew Linsley et al. “Tracking Without Re-recognition in Humans and Machines”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 19473–19486.
- [18] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).
- [19] William H Warren. “Information Is Where You Find It: Perception as an Ecologically Well-Posed Problem”. en. In: *Iperception* 12.2 (Mar. 2021), p. 20416695211000366.
- [20] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. en. In: *Nature* 518.7540 (Feb. 2015), pp. 529–533.
- [21] David Silver et al. “Mastering the game of Go without human knowledge”. en. In: *Nature* 550.7676 (Oct. 2017), pp. 354–359.
- [22] David Silver et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”. en. In: *Science* 362.6419 (Dec. 2018), pp. 1140–1144.
- [23] Oriol Vinyals et al. “Grandmaster level in StarCraft II using multi-agent reinforcement learning”. en. In: *Nature* 575.7782 (Nov. 2019), pp. 350–354.
- [24] Wu Li, Valentin Piëch, and Charles D Gilbert. “Perceptual learning and top-down influences in primary visual cortex”. en. In: *Nat. Neurosci.* 7.6 (June 2004), pp. 651–657.
- [25] Lisa S Scott, Olivier Pascalis, and Charles A Nelson. “A Domain-General Theory of the Development of Perceptual Discrimination”. en. In: *Curr. Dir. Psychol. Sci.* 16.4 (2007), pp. 197–201.
- [26] Pieter R Roelfsema, Arjen van Ooyen, and Takeo Watanabe. “Perceptual learning rules based on reinforcers and attention”. en. In: *Trends Cogn. Sci.* 14.2 (Feb. 2010), pp. 64–71.
- [27] L L Emberson. “Chapter One - How Does Experience Shape Early Development? Considering the Role of Top-Down Mechanisms”. In: *Advances in Child Development and Behavior*. Ed. by Janette B Benson. Vol. 52. JAI, Jan. 2017, pp. 1–41.
- [28] T N Wiesel and D H Hubel. “SINGLE-CELL RESPONSES IN STRIATE CORTEX OF KITTENS DEPRIVED OF VISION IN ONE EYE”. en. In: *J. Neurophysiol.* 26 (Nov. 1963), pp. 1003–1017.
- [29] T Watanabe, J E Náñez, and Y Sasaki. “Perceptual learning without perception”. en. In: *Nature* 413.6858 (Oct. 2001), pp. 844–848.
- [30] Lucas Lehnert, Michael L Littman, and Michael J Frank. “Reward-predictive representations generalize across tasks in reinforcement learning”. en. In: *PLoS Comput. Biol.* 16.10 (Oct. 2020), e1008317.
- [31] R C O’Reilly. “Generalization in interactive networks: the benefits of inhibitory competition and Hebbian learning”. en. In: *Neural Comput.* 13.6 (June 2001), pp. 1199–1241.

- [32] Kevin Frans et al. “Meta learning shared hierarchies”. In: *International Conference on Learning Representations*. 2018. arXiv: 1710.09767 [cs.LG]. URL: <https://openreview.net/forum?id=SyX0IeWAW>.
- [33] Zhongwen Xu, Hado P van Hasselt, and David Silver. “Meta-Gradient Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by S Bengio et al. Vol. 31. Curran Associates, Inc., 2018.
- [34] Zhongwen Xu et al. “Meta-gradient reinforcement learning with an objective discovered online”. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS’20 Article 1279. Vancouver, BC, Canada: Curran Associates Inc., Dec. 2020, pp. 15254–15264.
- [35] Rein Houthoofd et al. “Evolved Policy Gradients”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: <https://proceedings.neurips.cc/paper/2018/file/7876acb66640bad41f1e1371ef30c180-Paper.pdf>.
- [36] Abhishek Gupta et al. “Meta-Reinforcement Learning of Structured Exploration Strategies”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: <https://proceedings.neurips.cc/paper/2018/file/4de754248c196c85ee4fbdcee89179bd-Paper.pdf>.
- [37] Veronica Chelu, Doina Precup, and Hado van Hasselt. “Forethought and hindsight in credit assignment”. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS’20 Article 191. Vancouver, BC, Canada: Curran Associates Inc., Dec. 2020, pp. 2270–2281.
- [38] Vitchyr H Pong et al. “Offline Meta-Reinforcement Learning with Online Self-Supervision”. In: (July 2021). arXiv: 2107.03974 [cs.LG].
- [39] Karl Cobbe et al. “Quantifying Generalization in Reinforcement Learning”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1282–1289.
- [40] Mohit Vaishnav et al. “Understanding the Computational Demands Underlying Visual Reasoning”. en. In: *Neural Comput.* 34.5 (Apr. 2022), pp. 1075–1099.
- [41] Junkyung Kim, Matthew Ricci, and Thomas Serre. “Not-So-CLEVR: learning same-different relations strains feedforward neural networks”. en. In: *Interface Focus* 8.4 (Aug. 2018), p. 20180011.
- [42] B F Skinner. *The Behavior of Organisms: An Experimental Analysis*. en. D. Appleton & Company, Dec. 1938.
- [43] Max Jaderberg et al. “Reinforcement learning with unsupervised auxiliary tasks”. In: (Nov. 2016). arXiv: 1611.05397 [cs.LG].

- [44] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. “CURL: Contrastive Unsupervised Representations for Reinforcement Learning”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé Iii and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 5639–5650.
- [45] Evan Shelhamer et al. “Loss is its own Reward: Self-Supervision for Reinforcement Learning”. Feb. 2017.
- [46] Adam Stooke et al. “Decoupling Representation Learning from Reinforcement Learning”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 9870–9879.
- [47] Andrea Dittadi et al. “The Role of Pretrained Representations for the OOD Generalization of Reinforcement Learning Agents”. In: (July 2021). arXiv: 2107.05686 [cs.LG].
- [48] Amy Zhang et al. “Learning Invariant Representations for Reinforcement Learning without Reconstruction”. In: (June 2020). arXiv: 2006.10742 [cs.LG].
- [49] Yuval Tassa et al. “DeepMind Control Suite”. In: (Jan. 2018). arXiv: 1801.00690 [cs.AI].
- [50] Guy Davidson and Brenden M Lake. “Investigating Simple Object Representations in Model-Free Deep Reinforcement Learning”. In: (Feb. 2020). arXiv: 2002.06703 [cs.LG].
- [51] Drew Linsley et al. “Global-and-local attention networks for visual recognition”. In: (May 2018). arXiv: 1805.08819 [cs.CV].
- [52] Junkyung Kim* et al. “Disentangling neural mechanisms for perceptual grouping”. In: *International Conference on Representation Learning* (2020).
- [53] Yi Tay et al. “Long Range Arena: A Benchmark for Efficient Transformers”. In: *ICLR* (2021).
- [54] Drew Linsley et al. “Tracking Without Re-recognition in Humans and Machines”. In: (May 2021). arXiv: 2105.13351 [cs.CV].
- [55] Robert Geirhos et al. “Partial success in closing the gap between human and machine vision”. In: (June 2021). arXiv: 2106.07411 [cs.CV].
- [56] Robert Geirhos et al. “Generalisation in humans and deep neural networks”. In: *Advances in Neural Information Processing Systems 31*. Ed. by S Bengio et al. Curran Associates, Inc., 2018, pp. 7549–7561.
- [57] François Fleuret et al. “Comparing machines and humans on a visual categorization test”. en. In: *Proc. Natl. Acad. Sci. U. S. A.* 108.43 (Oct. 2011), pp. 17621–17625.
- [58] Christina M Funke et al. “Five points to check when comparing visual perception in humans and machines”. en. In: *J. Vis.* 21.3 (Mar. 2021), p. 16.
- [59] Drew Linsley et al. “Stable and expressive recurrent vision models”. In: *NeurIPS* (2020).
- [60] Drew Linsley et al. “Learning what and where to attend”. In: *International Conference on Learning Representations*. 2019.

- [61] Drew Linsley et al. “Learning long-range spatial dependencies with horizontal gated-recurrent units”. In: (May 2018). arXiv: 1805.08315 [cs.CV].
- [62] Matthew E. Taylor and Peter Stone. “Transfer Learning for Reinforcement Learning Domains: A Survey”. In: *Journal of Machine Learning Research* 10.56 (2009), pp. 1633–1685. URL: <http://jmlr.org/papers/v10/taylor09a.html>.
- [63] Rishabh Agarwal et al. “Deep Reinforcement Learning at the Edge of the Statistical Precipice”. In: (Aug. 2021). arXiv: 2108.13264 [cs.LG].
- [64] Robert Kirk et al. “A Survey of Generalisation in Deep Reinforcement Learning”. In: (Nov. 2021). arXiv: 2111.09794 [cs.LG].
- [65] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: (Dec. 2015). arXiv: 1512.03385 [cs.CV].
- [66] John Schulman et al. “Proximal Policy Optimization Algorithms”. In: (July 2017). arXiv: 1707.06347 [cs.LG].
- [67] Karl W Cobbe et al. “Phasic Policy Gradient”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 2020–2027.
- [68] P R Roelfsema, V A Lamme, and H Spekreijse. “The implementation of visual routines”. en. In: *Vision Res.* 40.10-12 (2000), pp. 1385–1411.
- [69] Pieter R Roelfsema. “Cortical algorithms for perceptual grouping”. en. In: *Annu. Rev. Neurosci.* 29 (Jan. 2006), pp. 203–227.
- [70] S Ullman. “Visual routines”. en. In: *Cognition* 18.1-3 (Dec. 1984), pp. 97–159.
- [71] W H Warren Jr et al. “Optic flow is used to control human walking”. en. In: *Nat. Neurosci.* 4.2 (Feb. 2001), pp. 213–216.
- [72] Runtao Liu et al. “The emergence of objectness: Learning zero-shot segmentation from videos”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 13137–13152.
- [73] Tsung-Yi Lin et al. “Feature Pyramid Networks for Object Detection”. In: (Dec. 2016). arXiv: 1612.03144 [cs.CV].
- [74] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.
- [75] Diederik P Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: (Dec. 2014). arXiv: 1412.6980 [cs.LG].
- [76] Alec Radford et al. “Learning Transferable Visual Models From Natural Language Supervision”. In: (Feb. 2021). arXiv: 2103.00020 [cs.CV].
- [77] Eric Wiewiora. “Reward Shaping”. In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I Webb. Boston, MA: Springer US, 2010, pp. 863–865.

- [78] Marc Bellemare et al. “Unifying Count-Based Exploration and Intrinsic Motivation”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. URL: <https://proceedings.neurips.cc/paper/2016/file/afda332245e2af431fb7b672a68b659d-Paper.pdf>.
- [79] Adrien Ecoffet et al. “First return, then explore”. en. In: *Nature* 590.7847 (Feb. 2021), pp. 580–586.
- [80] Adrien Ecoffet et al. “Go-Explore: a New Approach for Hard-Exploration Problems”. In: (Jan. 2019). arXiv: 1901.10995 [cs.LG].
- [81] Melrose Roderick, Christopher Grimm, and Stefanie Tellex. “Deep Abstract Q-Networks”. In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS ’18. Stockholm, Sweden: International Foundation for Autonomous Agents and Multiagent Systems, July 2018, pp. 131–138.
- [82] Tim Salimans and Richard Chen. “Learning Montezuma’s Revenge from a Single Demonstration”. In: (Dec. 2018). arXiv: 1812.03381 [cs.LG].
- [83] Lasse Espeholt et al. “IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1407–1416.

Chapter 5

Attractor Dynamics for Information Routing

5.1 Introduction

In the chapters prior, we worked through conceptualizations that merged an attractor dynamics viewpoint with that of large-scale deep recurrent neural networks. We saw how this computational construct can be beneficial for both the machine learning community and for cognitive scientists modeling human behavior. However, we had operated on mathematical and functional abstractions of these concepts without particularly touching on the notion of biological implementational plausibility. Does the brain construct and use attractors? If yes, then what are the supporting mechanisms? One of the primary goals of this chapter is to look into an implementation-level theory of how biological networks orchestrate attractor dynamics.

A growing body of evidence suggests that activity patterns in different regions of the brain can indeed resemble attractors. Among the early discoveries was the identification of persistent neural states (point attractors) that were hypothesized to serve as memory tokens [1, 2]. Persistence either manifested as consistent population firing rates or stable patterns of neural correlations. Slow ramping of persistent activity was later identified as a dynamical motif implementing oculomotor control in decision-making and evidence-accumulation tasks [3]. One of the striking examples of biological systems implementing attractor dynamics was in the case of head-direction cells [4]. These cells exhibited selectivity to an organism’s heading direction and markedly maintained persistent activity during stationary head states [5]. Moreover, they appeared to be topologically organized (akin to ring attractors), suggesting that attractor mechanisms also played a critical part in path integration and, ultimately, spatial navigation. The phenomenon of *bistability*, wherein cortical activity switches back and forth between two states, is another elucidating example of attractor dynamics in the brain [6]. Switches between states in bistable and multistable networks could be driven by either external inputs or internal stochasticities suggesting that the autonomicity of brain dynamics was context-dependent.

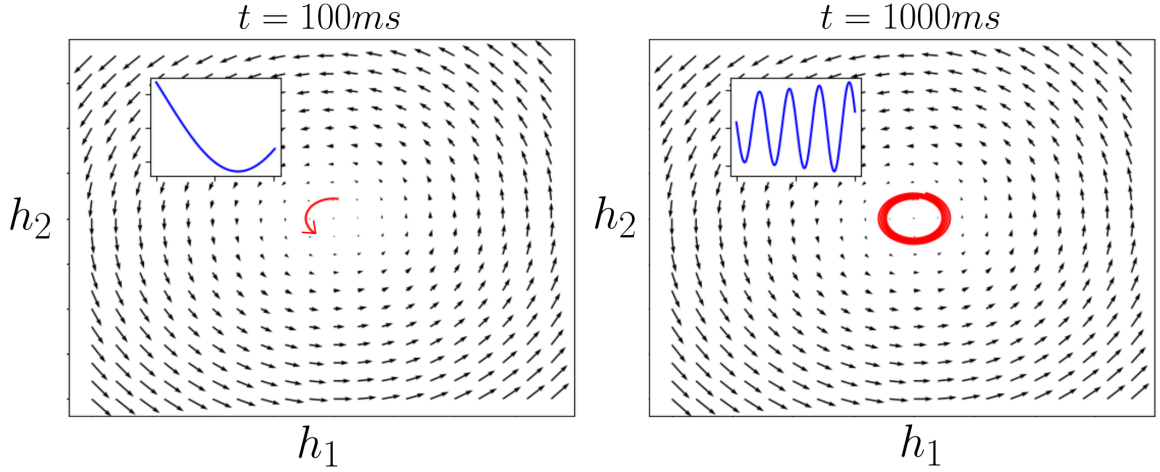


Figure 5.1: **The expressivity of an hGRU module.** The hGRU circuit implements Dale’s law and separates neurons into distinct excitatory (h_2) and inhibitory (h_1) sub-populations. It’s learned connectivity structure can allow for recurrent excitation and feedback inhibition. Here we show an example of how a low-dimensional activity manifold of the hGRU can resemble a quasi-stable oscillator. The network parameters driving these dynamics were identified using an approximate Bayesian inference algorithm [10].

Central pattern generators in the spinal cord [7] and working memory networks in the frontal cortex [8, 9] showed more complex attractor patterns such as limit cycles and oscillations. With recent advances in experimental neuroscience, we can now procure large-scale electrophysiological measurements and undoubtedly pinpoint the existence of such dynamical patterns in neural activity.

Identifying the mechanisms underpinning the formation of non-trivial attractor states has been a longstanding goal in computational and theoretical neuroscience. The dogma in this literature was based on four determining factors: (1) a network’s connectivity structure, (2) the associated connection weights, (3) distinct excitatory and inhibitory neuron contributions, and (4) non-linear single neuron properties. While large networks of non-linear neurons that were strongly and asymmetrically interconnected were prone to exhibit chaotic dynamics [11, 12], large networks with symmetric weights supported the emergence of point attractors [13]. In these networks, sufficient asymmetries in connectivity were achieved by simply constraining excitatory and inhibitory synapses to originate from distinct neurons as prescribed by Dale’s law [11, 14]. Asymmetric networks were also suggested to be responsible for periodic and quasi-periodic activity patterns [15]. In general, it was widely accepted that strong recurrent excitation counteracted the temporal decay of firing rates in neural populations and supported persistent activity [3]. Additionally, feedback inhibition was proposed to mitigate runaway excitation and stabilize networks with strong recurrent connectivity [16–18]. The temporal dynamics of excitation and inhibition also played a crucial part in network function. Rapid excitation mediated by AMPA receptors supported transient elevations in firing rates. Slow excitatory reverberations mediated by NMDA receptors supported the integration of information over longer periods of time [3]. Inhibition, both fast and slow, supports network stability and the

construction of “fixed points”.

Interestingly, our feedback architecture, the horizontal gated recurrent unit (hGRU), used throughout this thesis obeys all the four aforementioned principles for constructing attractors. The hGRU circuit implements Dale’s law and separates neurons into distinct excitatory and inhibitory sub-populations. Its architecture allows for recurrent excitation and feedback inhibition. And neurons in the hGRU circuit have non-linear response properties. We have shown in Chapter 3 that this circuit is indeed capable of exhibiting simple attractor patterns such as fixed points. We performed further numerical simulation experiments on a low-dimensional version of the circuit to determine its full range of neural dynamics. One such example is presented in Fig. 5.1. We observe complex attractor patterns such as quasi-stable oscillations when the network is inhibition-dominated, suggesting that the repertoire of computations the hGRU can carry out through its dynamics is expansive.

5.2 Activity-silent mechanisms and its affordances

One of the major shortcomings of the mechanistic dogma we reviewed above was the disparity in the time scales that its criteria demanded. Consider a network’s connectivity structure. In biological systems, connectivity patterns are shaped by synaptic plasticity induced by learning. Experience-driven modification of a network’s structure is a *slow* process that operates on timescales ranging from a few minutes to years [19]. The time courses of excitatory and inhibitory drives, and the subsequent modification of neural activity, on the other hand, are rather spontaneous fluctuations on the order of tens of milliseconds. Though there are affordances for slower conduction of excitatory currents, such as those mediated by NMDA receptors [3], they rather need to be carefully coordinated. The neural correlates of attractors in the brain exhibit both flexibility, in being able to reconfigure and adapt quickly (indicating that structural reconfiguration is likely *not* the driving factor), and stability, in being resistant to transient stochasticities (indicating the presence of components that moderate spontaneous neural activity). This called for introspection into possible neural mechanisms that can span a wide range of timescales outside of the slow-fast dichotomy.

Decoding efforts that aimed to interpret the attractor states in the brain put forth yet another consideration. For a very long time, it was believed that attractor states must be apparent in neural *activity* patterns. This belief was questioned by studies that discovered that attractors present at one point in time can be “lost” temporarily and reinstated later [20–23]. The notion of reinstating attractors from noise patterns *per se* was not new. For instance, Hopfield networks could converge to previously learned attractor points when initialized with unstructured noise. However, the attractors in [20–23] were not previously “learned” through multiple exposures but were simply constructed on-the-fly following the onset of a visual stimulus. This adaptive behavior pointed to the existence of mechanisms that operated faster than synaptic plasticity while being more stable compared to instantaneous fluctuations in activity. Neither the traditional models of attractor dynamics nor our learnable hGRU circuit accounted for this finding.

On a parallel note, neurobiological studies of synaptic transmission identified calcium as an

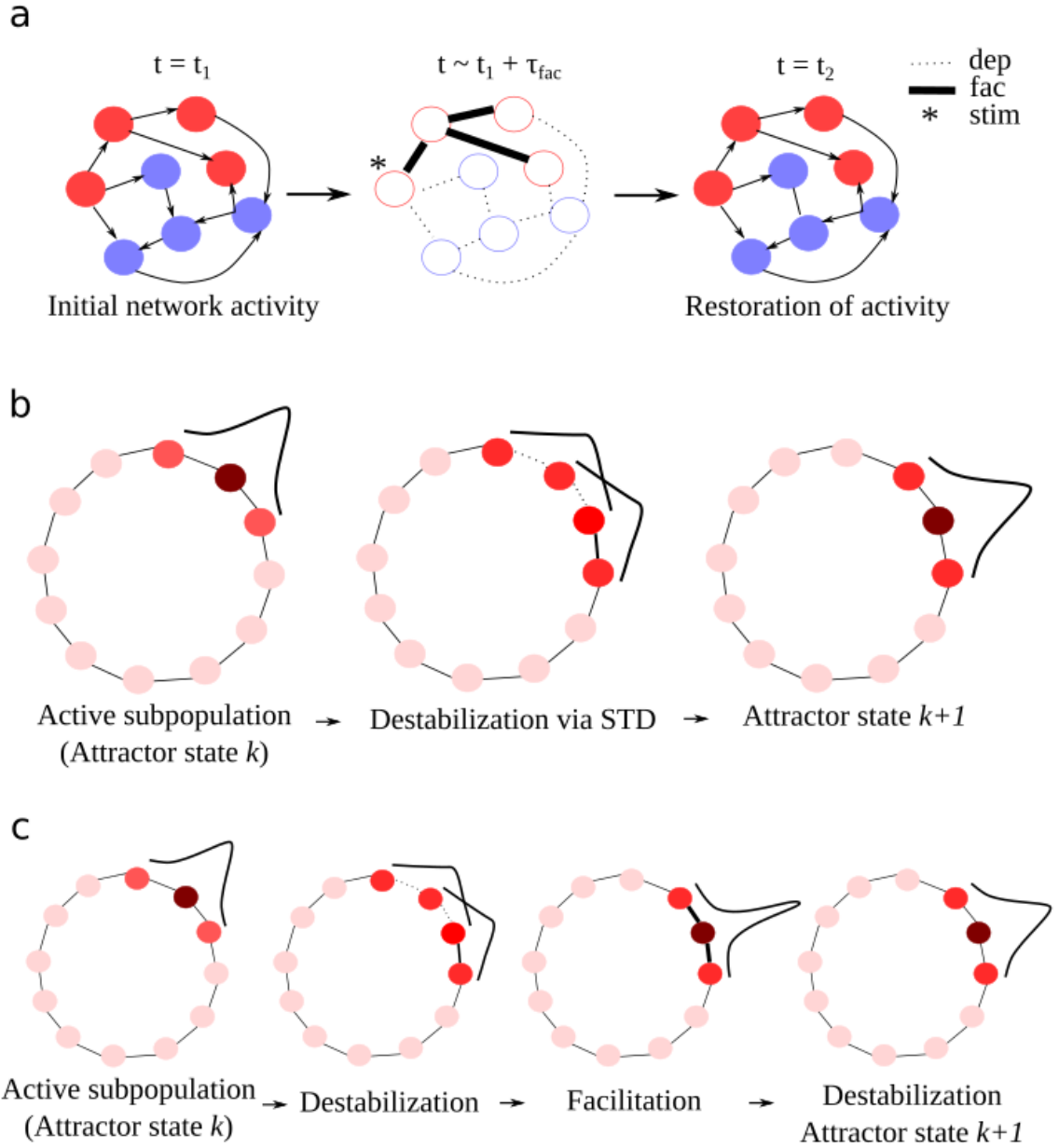


Figure 5.2: **Illustrating some of the computational benefits of dynamic synapses.** (a) Synaptic potentiation can maintain a “trace” of a network’s input even in the absence of overall activity and can subsequently be recovered through a non-specific probe. (b) Synaptic depression can suppress activity in a sub-network postsynaptic transmission and, in the process, generate a traveling wave-like phenomenon. (c) Similarly, strong facilitation following destabilization can generate damped oscillatory patterns.

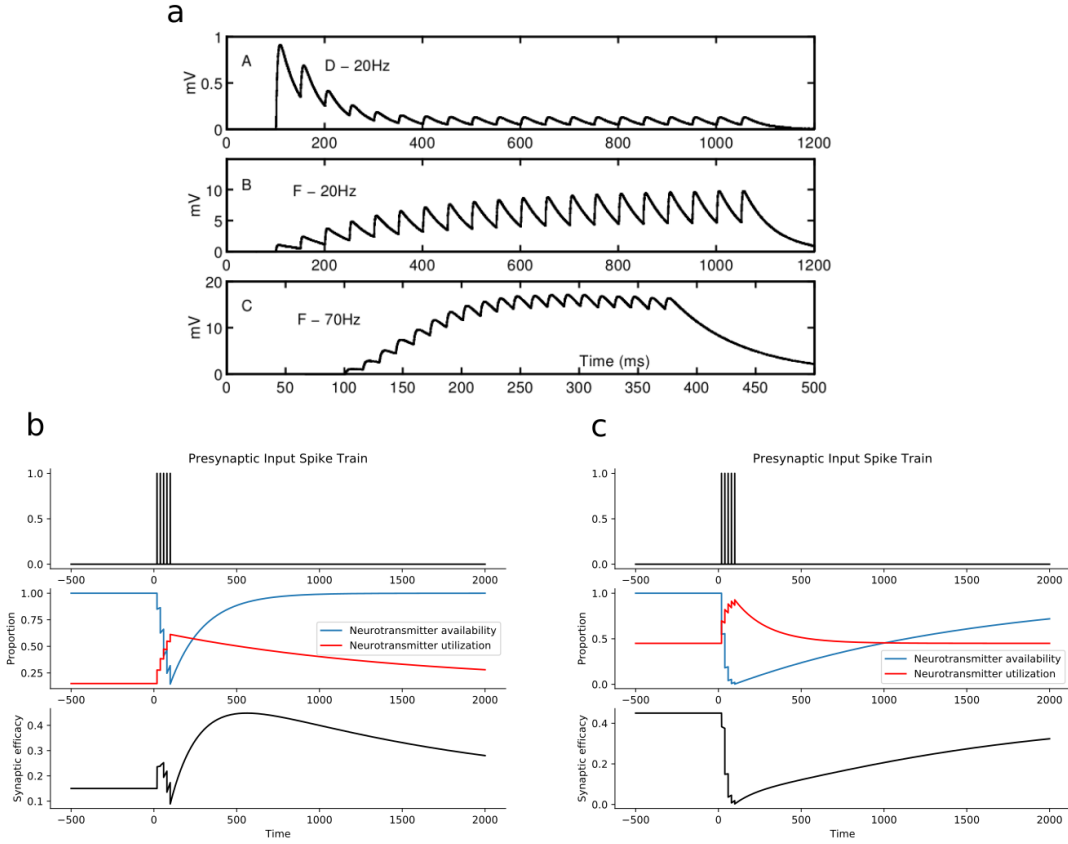


Figure 5.3: **Temporal filtering as an emergent property of dynamic synapses.** (a) (Top) Postsynaptic potential from a 20Hz stimulation through a depressing synapse. Subsequent volleys of presynaptic input evoke diminishing postsynaptic responses. (Middle) Postsynaptic potential from a 20Hz stimulation but through a facilitating synapse. Here, one can see a supra-linear effect of superposing volleys of presynaptic activity. (Bottom) Postsynaptic potential from a 70Hz stimulation through a facilitating synapse. Figure adapted from [29]. (b, c) Simulating synaptic efficacy (facilitation and depression, respectively) as per the model proposed by [30].

important regulator of synaptic strength [24, 25]. The temporary influx of calcium into the axon terminal following an action potential increased the probability of neurotransmitter release transiently until the residual calcium was cleared from the synaptic cleft. Such modulation of a synapse's net transmission ability became known as short-term synaptic plasticity (STSP). STSP lasted anywhere from a few milliseconds to a few minutes, regulated the flow of neural activity, and had several consequences for overall network function [26–28].

STSP emerged as a natural candidate mechanism for explaining activity-silent attractor dynamics. First, its time signatures matched the desired intermediate range between spike dynamics and synaptic plasticity dynamics. Second, the rate of calcium influx and subsequent clearance could combine to specify not one but multiple timescales – an important consideration for being adaptive. Third, the computational potential of networks with synapses modulated by STSP was immense [28].

Before we move on to examine some of these interesting properties, we make a note that STSP can manifest in two forms: facilitation and depression. In facilitating synapses, residual calcium in axon terminals persists following a first action potential and triggers more calcium release by subsequent action potentials and thereby producing large positive changes in synaptic strength [31, 32]. In depressing synapses, calcium influx is sensed by calcium-binding proteins that inhibit calcium channels, reducing further influx and ultimately restricting neurotransmitter release [33].

Over the years, several efforts have laid the groundwork for designing phenomenological models of STSP [26, 27, 29, 30]. The Tsodyks-Markram (TM) model, particularly, gained traction due to its elegant and parsimonious formulation. The TM model conceptualized synapses as finite-resource, autonomous entities. Specifically, a presynaptic action potential, at time t_* “activates” a fraction U of the total available synaptic resources. These activated resources, however, deactivate over time and recover back to baseline levels. Let τ_{in} and τ_{rec} be the time constants governing resource deactivation and recovery, and x, y, z represent the fraction of synaptic resources that are present in the recovered, active and inactive states, respectively. The dynamics of a synapse can then be specified by this coupled differential system.

$$\frac{dx}{dt} = \frac{z}{\tau_{rec}} - U \times x(t_*) \times \delta(t - t_*) \quad (5.1)$$

$$\frac{dy}{dt} = \frac{-y}{\tau_{in}} + U \times x(t_*) \times \delta(t - t_*) \quad (5.2)$$

$$\frac{dz}{dt} = \frac{y}{\tau_{in}} - \frac{z}{\tau_{rec}} \quad (5.3)$$

In this model, the fraction of synaptic resources in the recovered state (x) increases with the gradual recovery of inactive resources and decreases with further activation caused by subsequent action potentials (Eq. 5.1). Similarly, the fraction of active synaptic resources decreases with rapid inactivation and increases following activation by an action potential (Eq. 5.2). The fraction of inactive synaptic resources increases with the inactivation of active resources and decreases with recovery from an inactive state (Eq. 5.3). Essentially, the interplay between x , y , and z here is a zero-sum game.

The dynamical system formulation above only accounts for synaptic depression and not facilitation. To capture facilitation, the TM model considers U as a variable quantity that gets incremented by each subsequent action potential. By denoting the current state of U with $\hat{U}$ and the time constant for facilitation with τ_{fac} , the TM model also couples the dynamics of $\hat{U}$ with that of the synaptic resources as follows.

$$\frac{d\hat{U}}{dt} = \frac{-\hat{U}}{\tau_{fac}} - U \times (1 - \hat{U}) \times \delta(t - t_{sp}) \quad (5.4)$$

The beauty of this formulation lies in the interdependence of facilitation and depression. The facilitation dynamics dictate the magnitude of U , which in turn determines the strength and temporal properties of depression. Discretizing Eq. 5.4, we can write the Euler update equation for $\hat{U}$:

$$\hat{U}[n+1] = U + \hat{U}[n] \times (1 - U) \times e^{\frac{-\delta t}{\tau_{fac}}} \quad (5.5)$$

In Eq. 5.5, δt denotes the time between the n^{th} and $(n + 1)^{th}$ action potential. Furthermore, by assuming a constant firing rate λ for the presynaptic neuron, we can derive the steady value of $\hat{U}$.

$$\hat{U}_{\infty} = \frac{U}{1 - (1 - U) \times e^{\frac{-1}{\lambda \tau_{fac}}}} \quad (5.6)$$

Short-term synaptic plasticity mechanisms regulate and balance neural activity in cortical networks and have provable computational benefits [34]. We review some of these known properties, suggest newer ones, and make the case that dynamic synapses must be an integral part of attractor networks.

Temporal filtering: From Eq. 5.6, we see that the activation of synaptic resources depends on the presynaptic firing rate λ . This formulation naturally affords temporal filtering. Consider a burst of action potentials arriving at a depression-dominant synapse as in Fig. 5.3(c). Neurotransmitter depletion occurs rapidly following the first few spikes, thus preventing further information transmission. Such a synapse thus functions as a low-pass filter optimized for communicating lower firing rates. On the other hand, a facilitation dominant synapse, as in Fig. 5.3(b), is optimized for high-frequency information transmission, i.e., a high pass filter. These effects are flexible and vary with the actual firing rate of the presynaptic neuron, as shown in Fig. 5.3(a). A postsynaptic neuron can also exhibit both high-pass and low-pass filtering properties, similar to those observed in midbrain electrosensory neurons when it receives inputs from both facilitating and depressing synapses [35]. Synaptic depression can also remove autocorrelations between input and output spikes [36], while facilitation can help increase the sensitivity to temporal correlations in the input [37].

Gain control: A neuron typically integrates hundreds of inputs, each of which spans a range of $1 - 100$ Hz. Despite the heterogeneity and volume of computations a neuron performs, it still manages to keep its activity in the $1 - 100$ Hz range. This is an effect called gain control [28]. As with temporal filtering, the role of the presynaptic firing rate in Eq. 5.6 supports gain control. If we consider a small, abrupt change in presynaptic firing rate from λ to $\lambda + \Delta\lambda$, then $\Delta\hat{U}$ is logarithmically scaled with respect to $\Delta\lambda$. This is a realization of Weber’s law.

Attractor dynamics: Short-term plasticity greatly enhances the repertoire of dynamical patterns an attractor network can exhibit. Most apparent is the role of synaptic efficacy as an information buffer (Fig. 5.2a). A synapse’s efficacy reflects recent patterns of presynaptic activity and is of a longer time scale than neural activity. Thus, traces of transient inputs can be maintained in the synaptic efficacies for short amounts of time even in the absence of any network activity [30, 38]. Another interesting emergent property in networks with STSP is that of a traveling wave (Fig. 5.2b,c). In networks with continuous attractors, short-term synaptic depression can destabilize an attractor state and nudge the system into one of many competing states [39, 40]. This phenomenon is also known as “directed drift” [40]. The concept of systems transitioning between attractors states is well documented in neuroscience and has important ramifications for cognitive processes such as working memory [9]. Though recurrently connected networks without STSP can also exhibit attractor

drifts [41], they could only account for constant-velocity traveling waves. In networks with synaptic depression, this was not a concern. The downside of destabilization by short-term depression is degradation of the attractor state itself over time. Short-term facilitation, on the other hand, can work to counteract these side effects, reduce directed drift, and help stabilize attractor states [42]. In Fig. 5.2c, we present a scheme in which a ring network augmented with both facilitation and depression can alternate (as per Eq. 5.6 and 5.3) to produce varying-amplitude traveling waves. *In essence, STSP provides a unifying implementational-level hypothesis for disparate network-level phenomenon like neural oscillations, synchrony, activity-silent states, and attractor dynamics.*

5.3 Synaptic modulation in feedback networks

Despite the obvious computational advantages conferred by short-term plasticity mechanisms, to date, there have been no models probing either their role or utility for behavior on complex visual challenges. One primary reason for this is the notoriety associated with training recurrent neural networks (RNNs), as we saw in Chapter 3. Moreover, synapses modulated by STSP are autonomous dynamical systems themselves, and hence an RNN with dynamic synapses can be thought of as a “system of systems”, making the training problem even harder. In fact, learnable RNN models that successfully employed dynamic synapses have either relied on highly-constrained, low-dimensional tasks [38] or coarse functional approximations of synaptic modulation [43–45].

Recent advances in the machine learning sphere, in the context of empirical scaling laws, indicate that the conclusions one can draw from a system’s behavior can dramatically change with an increase in its task- and parameter-complexity [46–49]. It is thus vital that we evaluate theories of synaptic mechanisms in the context of networks performing complex behaviors. In this effort, we adopt a hybrid approach toward scaling up with three specific goals (Fig. 5.4a). First, we wanted to propose a fully-trainable architecture that was image-computable and leveraged the power of both deep feature extractors and recurrent neural dynamics. Second, we wanted our framework to support both the extraction of complex visual representations and their appropriate utilization. This meant that our system had to, for the first time, reconcile mechanisms of visual representation learning and executive processing. Third, we wanted to test if the hypothesized role of fast synaptic modulation did indeed scale up to complex behaviors.

5.3.1 Methods

Choice of a visual task

We consider a classic visual working memory paradigm: the delayed match to (rotated) sample [50]. This task was challenging as it necessitated both the extraction of non-trivial visual features and their subsequent maintenance and manipulation. To mimic the perspective line drawings from Shepherd and Metzler [51], we parametrically generated 3D-polyomino stimuli (using custom code written in Blender) that varied in arm lengths, in-plane, and out-of-plane rotations. Each trial proceeded as

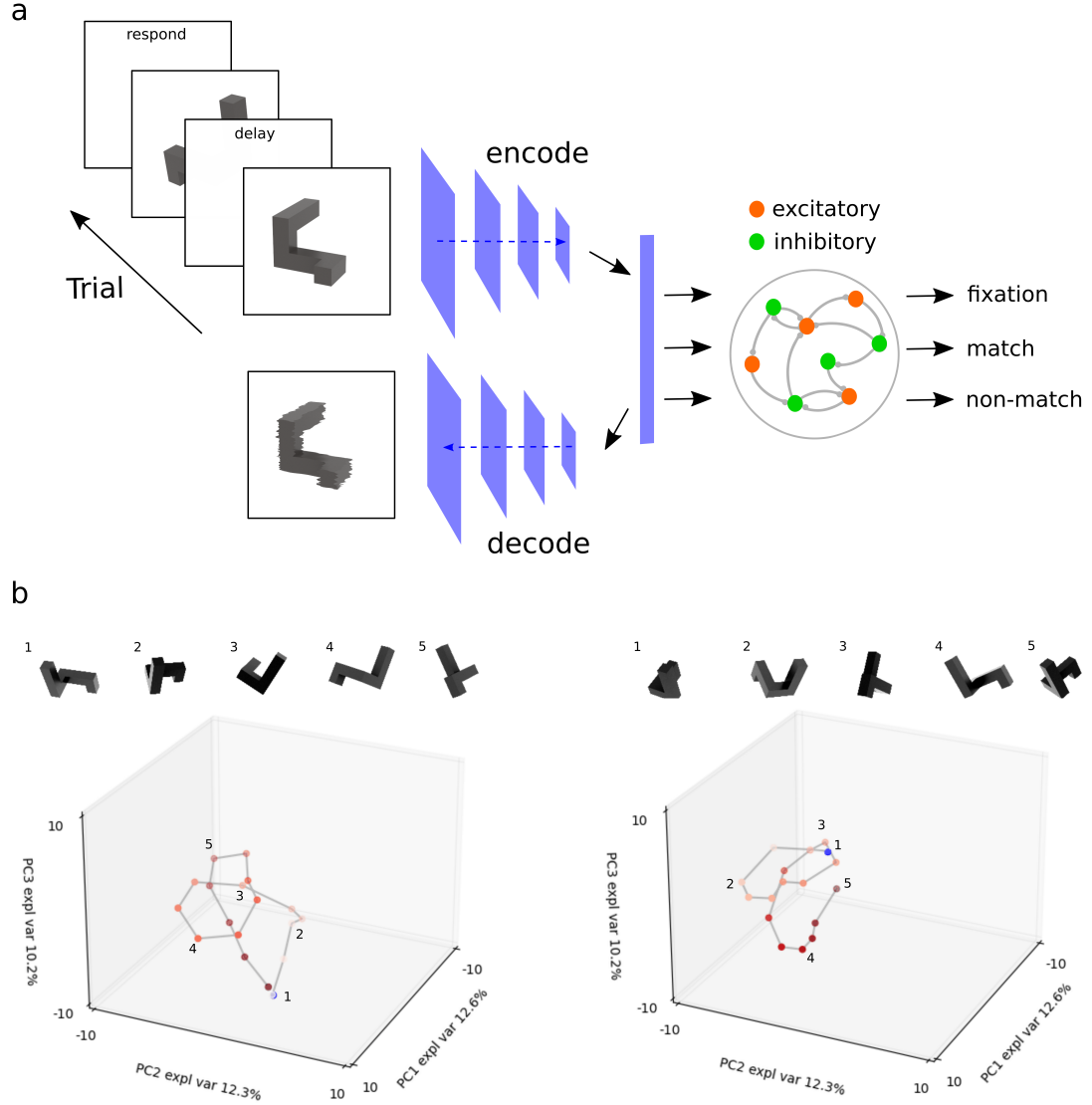


Figure 5.4: **A hybrid, image-computable model of visual manipulation.** (a) A ResNet-18 encoder-decoder architecture is trained from scratch to extract latent visual representations from the input shapes. These representations are then fed into an RNN with synapses modulated by STSP. The RNN is tasked with determining if there was indeed a match between the sample and test objects presented on either sides of a delay period. Though the whole architecture is trainable end-to-end, in this study we separate the encoder-decoder training from that of the RNN. (b) Visualizing the latent embeddings from the encoder reveals its smooth but highly non-linear structure. Here we depict how variations in object viewpoint are encoded in the principle components of the latent encoding.

follows. After an initial $500ms$ fixation period a sample object was presented for $500ms$. Following this, there was a delay period of $1000ms$. Finally, a test object was presented for $500ms$ at the end of which the model was required to determine if the sample and test objects were a match. We note

that typical working memory models use “ideal observer” neural tuning as inputs and are not image computable [30, 38]. As we alluded to before, the problem with using idealized tuning is that it doesn’t scale to complex stimuli, such as the polyominoes used in this study. Moreover, if idealized neural encodings are equivariant to transformations of the stimuli, then the challenge of stimulus “manipulation” becomes trivial.

Network model

We adopt a hybrid modeling approach for this task (Fig. 5.4a). For the vision front-end, we employ a state-of-the-art deep convolutional architecture, the ResNet-18 [52], that extracts compressed latent representations from input frames of dimensions 128×128 px. To provide training signal for the encoder, we additionally use a ResNet-18 *decoder* that reconstructs input frames (per time point) from the latent encodings. We use a latent embedding size $z = 128$ for the encoder. The encoder-decoder architecture is trained from scratch for 100 epochs using Stochastic Gradient Descent implemented by the Adam optimizer ($\text{lr}=1e-5$) [53], to minimize the reconstruction loss.

The latent embedding extracted from the ResNet-18 encoder serves as the feedforward drive for the RNN that is tasked with solving the delayed match to sample trial. We follow the recipe of [38], with some modifications, to construct an RNN that can learn to transform time-varying object representations into match-nonmatch decisions. Our RNN consisted of $N = 128$ neurons with recurrent connectivity given by the kernel $W_{rec} \in \mathbb{R}^{128 \times 128}$. Similar to [38], 80% of neurons in our RNN were excitatory and 20% inhibitory. $W_{in} \in \mathbb{R}^{z \times 128}$ projected the visual embeddings onto the recurrent population at each time point. The synapses in our RNN were governed by short-term plasticity rules as per the conventions of [30, 38]. They obeyed the following differential system,

$$\frac{dx(t)}{dt} = \frac{1 - x(t)}{\tau_x} - u(t) \times x(t) \times r(t) \times \Delta t, \quad (5.7)$$

$$\frac{du(t)}{dt} = \frac{U - u(t)}{\tau_u} + U \times (1 - u(t)) \times r(t) \times \Delta t, \quad (5.8)$$

where x and u represented the fraction of synaptic resources that were available and utilized, respectively. τ_x and τ_u governed the temporal dynamics of neurotransmitter recovery and utilization. $r(t)$ represented the presynaptic firing rate and Δt the membrane time constant. A synapse was facilitating when $\tau_x < \tau_u$, and depressing otherwise. For our experiments, we set $\tau_x = 1500ms$ and $\tau_u = 200ms$ for depressing synapses and vice versa for facilitating synapses. 50% of our recurrent synapses were facilitating and 50% were depressing. Overall, the recurrent dynamics obeyed the following dynamical system:

$$\tau \frac{d\mathbf{r}(t)}{dt} = -\mathbf{r} + \left[\underbrace{W_{rec} \mathbf{x}(t) \mathbf{u}(t)}_{\text{efficacy}} \mathbf{r}(t) + \underbrace{W_{in} \mathbf{u}(t)}_{\text{feedforward drive}} + \mathbf{b}_{rec} + \underbrace{\xi(t)}_{\text{noise}} \right]_+ \quad (5.9)$$

A primary difference between our formulation and that of [38] is that we allowed for each synapse in the network to be individually modulated as opposed to all outgoing synapses from a neuron being

treated the same. We employ a linear readout function to transform hidden activity in the RNN to a decision. Specifically, the readout layer was given by,

$$\mathbf{y}(t) = \sigma(W_{out}\mathbf{r}(t) + \mathbf{b}_{out}), \quad (5.10)$$

where $\sigma(\cdot)$ is the softmax function, and $W_{out} \in \mathbb{R}^{3 \times z}$. The RNN was trained using backpropagation through time to minimize the Cross Entropy Loss during the “test” timepoints.

5.3.2 Results

Stimulus representations are highly non-linear and entangled

Training the encoder-decoder architecture as a visual feature extractor affords image-computability and avoids the caveats of idealized responses. Our data-driven approach can scale to arbitrarily complex and naturalistic stimulus paradigms where “ideal” response construction is not obvious. Moreover, whilst studying the manipulation of visual representations, we wish to avoid scenarios where representations are linearly-equivariant to transformations of the input. For example, linear transformations of the idealized motion tuning of input neurons in [38] are non-identifiable and severely restricts the scope of scientific questions that can be asked.

We analyze the latent space of our encoder model to ascertain that *object-category* information is not trivially attainable from the object representation itself. We perform a dimensionality reduction using PCA [54] on the set of latent embeddings $\{\mathbf{z}_{(i)}\}_{i=1}^{N_s}$, where N_s denotes the total number of train samples, and visualize variations in the first three principal components to changes in object viewpoint. A representative example is shown in Fig. 5.4b. The encodings of the same object rotated along different principal axes reveals smooth but nonlinear perturbations in the representational space. This guarantees that an RNN trying to tease apart different viewpoints of the same object from different objects has to learn to traverse this complex space and in the process learn to perform non-trivial visual manipulation.

Population decoding reveals a deeper role for fast synaptic plasticity

Computational models investigating the role of short-term synaptic plasticity in working memory networks more or less converged to one dominant viewpoint: synaptic modulation is primarily responsible for the maintenance of information in activity-silent states [21, 23, 30, 38, 55]. Recently, [38] demonstrated that in a network trained to perform a delayed match to sample task, population decoding analysis can shed light on the informational content of different network substrates. Here, we revisit the lessons learned from the population decoding approach and seek to establish its relationship to task-complexity.

Specifically, we were interested in examining the differential roles of neural activity and synaptic efficacies in encoding information. We trained linear, multiclass support vector machines, at each time point separated 10ms, to classify the *object-identity* of the sample from both the instantaneous neural activity in a trained RNN as well as its synaptic efficacies (as in Eq. 5.9). We generated a

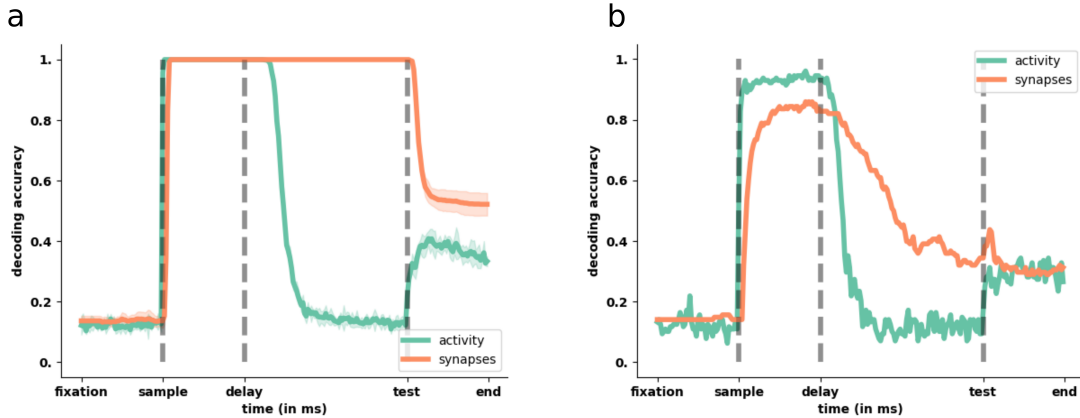


Figure 5.5: **Population decoding analysis on RNNs trained to solve the delayed match to sample task.** (a) Sample decoding accuracy from neural activity (in green) and synaptic efficacy (in orange) reveals that a model trained to perform a delayed match to sample task, on a restricted set of sample viewpoints, primarily utilizes synaptic substrates to store information about the sample over the delay period. (b) However, the same model when trained over a wide variety of sample viewpoints adopts a more complex strategy that does not obey a clear dichotomy. Both models successfully solve their respective tasks with accuracies $> 90\%$.

random batch of 1024 trials for the decoding analysis and separated samples from each time point into dedicated train (75%) and test (25%) partitions.

We first perform the decoding analysis on a model trained to solve the delayed match to sample task with the 3D polyominoes under one condition. This model was trained on trials where only one canonical viewpoint of each object (a unique 3D shape) was used. Population decoding results for this scenario are reported in Fig. 5.5a. We used a total of 6 unique objects implying that the baseline classification accuracy was ≈ 0.167 . Naturally, during the initial fixation period, decoding from both activity and synaptic efficacy was at baseline performance. Similarly, after the presentation of the sample, decoding from both activity and synaptic efficacy was near perfect. While information about the sample identity in neural activity decayed over the delay period, this information was still perfectly maintained in the synaptic efficacies. This suggested that sample information was reliably encoded in the synaptic efficacies and weakly encoded in the neuronal activities themselves over the delay period. The pattern we observed here was remarkably consistent with what has been reported in the literature [38].

We subsequently perform a similar decoding analysis on a RNN model trained over arbitrary object viewpoints. We clarify that the task here is still delayed match to sample, but trials in the training set were not confined to use only one canonical viewpoint per object. Only trials that had the same object in the same viewpoint configuration during both the sample and test presentations were considered a “match”. Our model was able to successfully solve the task with accuracies consistently over 90%. This suggested that the model was not only able to tell object templates apart, but also different configurations of the same object. Interestingly, the population decoding analysis for this

model did not subscribe to the dichotomy observed before (Fig. 5.5b). Despite the model successfully solving the same-different task, information about the sample did not seem to be neither explicitly nor exclusively maintained in any one neural substrate. This example is hence illuminating on two different fronts. First, this calls into question the reductionist assumption that synaptic modulation is primarily responsible for short-term storage and suggests a more dynamic role in information processing. Second, it reiterates the value of developing models that can scale. Depending on their training diet, neural network models are often susceptible to finding solutions that are not general, and interpreting model strategies needs careful introspection.

“Peeking into” the RNN’s decision making

While the decoding approach aims to tell us the information content of each model substrate, it does not shed light on *how* they are used. One of the many advantages of having an image-computable model is that we can visually decode its latent representations to reveal strategies that the model employs.

As we noted in Fig. 5.5b, both neuronal activity and synaptic efficacy encode sample-identity information reliably at the onset of sample presentation. We take advantage of this insight and assume that there is enough information in the synaptic efficacies during the sample phase to be able to visually reconstruct the “memory” using our pretrained ResNet-18 decoder network (Fig. 5.4a). Specifically, we hypothesize that there exists a linear mapping $g(\cdot)$ between the synaptic efficacy vector $\mathbf{e}_t^{(k)} = \sum_j \{\mathbf{x}^{(k)}(t)\mathbf{u}^{(k)}(t)\}_{i,j}$ and the embedding vector $\mathbf{z}^{(k)}$ for sample k at time t during the sample-presentation phase. Once trained, we can apply the transformation post hoc to other time points in the trial to visually probe the memory as $f_{dec}(g(\mathbf{e}_t^{(k)}))$, where f_{dec} is the pretrained ResNet-18 decoder. As with the neural population decoding, we generate a batch of 1024 random trials to train the linear mapping. An example output of this visual decoding process is shown in Fig. 5.6. We observe that while synaptic variables hold sample information over the delay period, the fidelity of these representations decay with time. Interestingly, this drop in fidelity did not seem to ultimately impact model performance in the same-different task. When we probe synaptic memory in the test phase, we find that remnants of sample features are still decodable. Taken together, these findings suggest that computational mechanisms far more complex than straightforward reactivation of potentiated synapses are at play in this setting.

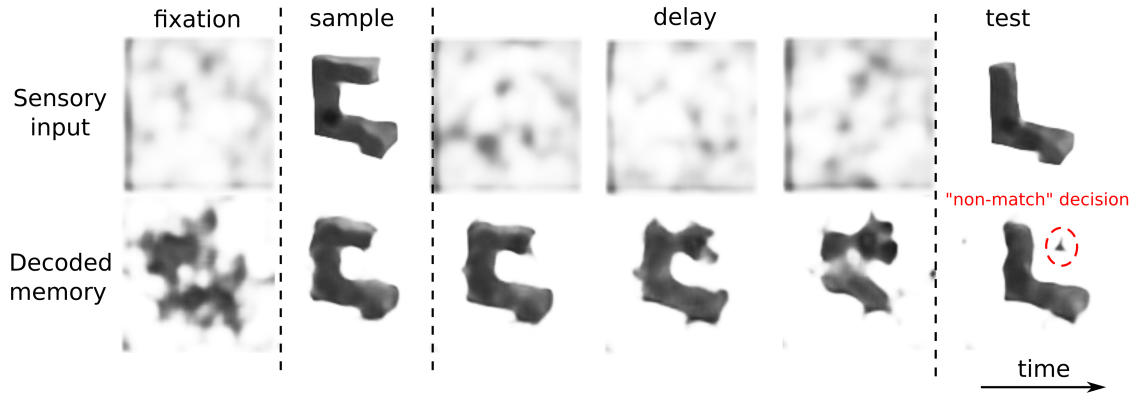


Figure 5.6: **Visual decoding of latent RNN representations over the delay period.** Information in the synaptic efficacies of a trained RNN, over the course of a trial, can be visually inspected by our pre-trained ResNet-18 decoder. To do so, we first train a linear map to project synaptic variables into the decoder’s embedding space and subsequently use this transformation to reconstruct the “memory” from synaptic traces. We present an illustrative example to highlight how visual decoding strategies can shed light onto the complex computational mechanisms employed by RNNs with dynamic synapses.

5.4 References

- [1] Shintaro Funahashi, Charles J Bruce, and Patricia S Goldman-Rakic. “Mnemonic coding of visual space in the monkey’s dorsolateral prefrontal cortex”. In: *Journal of neurophysiology* 61.2 (1989), pp. 331–349.
- [2] William A Little. “The existence of persistent states in the brain”. In: *Mathematical biosciences* 19.1-2 (1974), pp. 101–120.
- [3] Xiao-Jing Wang. “Probabilistic decision making by slow reverberation in cortical circuits”. In: *Neuron* 36.5 (2002), pp. 955–968.
- [4] James J Knierim, Hemant S Kudrimoti, and Bruce L McNaughton. “Place cells, head direction cells, and the learning of landmark stability”. In: *Journal of Neuroscience* 15.3 (1995), pp. 1648–1659.
- [5] Kechen Zhang. “Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: a theory”. In: *Journal of Neuroscience* 16.6 (1996), pp. 2112–2126.
- [6] Megan Wang, Daniel Arteaga, and Biyu J He. “Brain mechanisms for simple perception and bistable perception”. In: *Proceedings of the National Academy of Sciences* 110.35 (2013), E3350–E3359.
- [7] Eve Marder and Dirk Bucher. “Central pattern generators and the control of rhythmic movements”. In: *Current biology* 11.23 (2001), R986–R996.
- [8] André M Bastos et al. “Laminar recordings in frontal cortex suggest distinct layers for maintenance and control of working memory”. In: *Proceedings of the National Academy of Sciences* 115.5 (2018), pp. 1117–1122.
- [9] E. K. Miller, M. Lundqvist, and A. M. Bastos. “Working Memory 2.0”. In: *Neuron* 100.2 (Oct. 2018), pp. 463–475.
- [10] Alvaro Tejero-Cantero et al. “SBI-A toolkit for simulation-based inference”. In: *arXiv preprint arXiv:2007.09114* (2020).
- [11] Haim Sompolinsky, Andrea Crisanti, and Hans-Jurgen Sommers. “Chaos in random neural networks”. In: *Physical review letters* 61.3 (1988), p. 259.
- [12] Carl Van Vreeswijk and Haim Sompolinsky. “Chaos in neuronal networks with balanced excitatory and inhibitory activity”. In: *Science* 274.5293 (1996), pp. 1724–1726.
- [13] John J Hopfield. “Neural networks and physical systems with emergent collective computational abilities.” In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.
- [14] Henry Dale. *Pharmacology and nerve-endings*. 1935.
- [15] Haim Sompolinsky and Ido Kanter. “Temporal association in asymmetric neural networks”. In: *Physical review letters* 57.22 (1986), p. 2861.

- [16] Nicolas Brunel and Xiao-Jing Wang. “Effects of neuromodulation in a cortical network model of object working memory dominated by recurrent inhibition”. In: *Journal of computational neuroscience* 11.1 (2001), pp. 63–85.
- [17] Nicolas Brunel and Xiao-Jing Wang. “What determines the frequency of fast network oscillations with irregular neural discharges? I. Synaptic dynamics and excitation-inhibition balance”. In: *Journal of neurophysiology* 90.1 (2003), pp. 415–430.
- [18] Hirofumi Ozeki et al. “Inhibitory stabilization of the cortical network underlies visual surround suppression”. In: *Neuron* 62.4 (2009), pp. 578–592.
- [19] Guo-qiang Bi and Mu-ming Poo. “Synaptic modification by correlated activity: Hebb’s postulate revisited”. In: *Annual review of neuroscience* 24.1 (2001), pp. 139–166.
- [20] Dirk van Moorselaar et al. “Spatially selective alpha oscillations reveal moment-by-moment trade-offs between working memory and attention”. In: *Journal of cognitive neuroscience* 30.2 (2018), pp. 256–266.
- [21] Nathan S Rose et al. “Reactivation of latent working memories with transcranial magnetic stimulation”. In: *Science* 354.6316 (2016), pp. 1136–1139.
- [22] Michael J Wolff et al. “Revealing hidden states in visual working memory using electroencephalography”. In: *Frontiers in systems neuroscience* 9 (2015), p. 123.
- [23] Mark G Stokes. “‘Activity-silent’ working memory in prefrontal cortex: a dynamic coding framework”. In: *Trends in cognitive sciences* 19.7 (2015), pp. 394–405.
- [24] Neil R Hardingham et al. “Extracellular calcium regulates postsynaptic efficacy through group 1 metabotropic glutamate receptors”. In: *Journal of Neuroscience* 26.23 (2006), pp. 6337–6345.
- [25] William A Catterall and Alexandra P Few. “Calcium channel regulation and presynaptic plasticity”. In: *Neuron* 59.6 (2008), pp. 882–901.
- [26] Misha V Tsodyks and Henry Markram. “The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability”. In: *Proceedings of the national academy of sciences* 94.2 (1997), pp. 719–723.
- [27] Henry Markram and Misha Tsodyks. “Redistribution of synaptic efficacy between neocortical pyramidal neurons”. In: *Nature* 382.6594 (1996), pp. 807–810.
- [28] LF Abbott and Wade G Regehr. “Synaptic computation”. In: *Nature* 431.7010 (2004), pp. 796–803.
- [29] Misha Tsodyks, Klaus Pawelzik, and Henry Markram. “Neural networks with dynamic synapses”. In: *Neural computation* 10.4 (1998), pp. 821–835.
- [30] G. Mongillo, O. Barak, and M. Tsodyks. “Synaptic theory of working memory”. In: *Science* 319.5869 (Mar. 2008), pp. 1543–1546.
- [31] Bo Katz and R Miledi. “The role of calcium in neuromuscular facilitation”. In: *The Journal of physiology* 195.2 (1968), pp. 481–492.

- [32] Frederick A Dodge Jr and Rami Rahamimoff. “Co-operative action of calcium ions in transmitter release at the neuromuscular junction”. In: *The Journal of physiology* 193.2 (1967), pp. 419–432.
- [33] Michael A Gaviño et al. “Homeostatic synaptic depression is achieved through a regulated decrease in presynaptic calcium channel abundance”. In: *Elife* 4 (2015), e05473.
- [34] Jesus M Cortes et al. “Short-term synaptic plasticity in the deterministic Tsodyks–Markram model leads to unpredictable network dynamics”. In: *Proceedings of the National Academy of Sciences* 110.41 (2013), pp. 16610–16615.
- [35] Eric S Fortune and Gary J Rose. “Short-term synaptic plasticity as a temporal filter”. In: *Trends in neurosciences* 24.7 (2001), pp. 381–385.
- [36] Mark S Goldman, Pedro Maldonado, and LF Abbott. “Redundancy reduction and sustained firing with stochastic depressing synapses”. In: *Journal of Neuroscience* 22.2 (2002), pp. 584–591.
- [37] Mark A Bourjaily and Paul Miller. “Dynamic afferent synapses to decision-making networks improve performance in tasks requiring stimulus associations and discriminations”. In: *Journal of neurophysiology* 108.2 (2012), pp. 513–527.
- [38] N. Y. Masse et al. “Circuit mechanisms for the maintenance and manipulation of information in working memory”. In: *Nat. Neurosci.* 22.7 (July 2019), pp. 1159–1167.
- [39] Lawrence Christopher York and Mark CW Van Rossum. “Recurrent networks with short term synaptic depression”. In: *Journal of computational neuroscience* 27.3 (2009), pp. 607–620.
- [40] Sandro Romani and Misha Tsodyks. “Short-term plasticity based network model of place cells dynamics”. In: *Hippocampus* 25.1 (2015), pp. 94–105.
- [41] David Hansel and Haim Sompolinsky. *Modeling feature selectivity in local cortical circuits*. 1998.
- [42] Alexander Seeholzer, Moritz Deger, and Wulfram Gerstner. “Stability of working memory in continuous attractor networks under the control of short-term plasticity”. In: *PLoS computational biology* 15.4 (2019), e1006928.
- [43] Mathieu Chalvidal et al. “Go with the flow: Adaptive control for neural odes”. In: *arXiv preprint arXiv:2006.09545* (2020).
- [44] Thomas Miconi. “Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks”. In: *Elife* 6 (2017), e20899.
- [45] Thomas Miconi et al. “Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity”. In: *arXiv preprint arXiv:2002.10585* (2020).
- [46] Jared Kaplan et al. “Scaling laws for neural language models”. In: *arXiv preprint arXiv:2001.08361* (2020).

- [47] Ben Sorscher et al. “Beyond neural scaling laws: beating power law scaling via data pruning”. In: *arXiv preprint arXiv:2206.14486* (2022).
- [48] Jonathan S Rosenfeld. “Scaling laws for deep learning”. In: *arXiv preprint arXiv:2108.07686* (2021).
- [49] Yasaman Bahri et al. “Explaining neural scaling laws”. In: *arXiv preprint arXiv:2102.06701* (2021).
- [50] Thomas A Daniel, Jeffrey S Katz, and Jennifer L Robinson. “Delayed match-to-sample in working memory: A BrainMap meta-analysis”. In: *Biological Psychology* 120 (2016), pp. 10–20.
- [51] Roger N Shepard and Jacqueline Metzler. “Mental rotation of three-dimensional objects”. In: *Science* 171.3972 (1971), pp. 701–703.
- [52] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [53] D P Kingma and J Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [54] Michael E Tipping and Christopher M Bishop. “Probabilistic principal component analysis”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.3 (1999), pp. 611–622.
- [55] Michael J Wolff et al. “Dynamic hidden states underlying working-memory-guided behavior”. In: *Nature neuroscience* 20.6 (2017), pp. 864–871.

Chapter 6

Discussion and Conclusion

The second coming of deep neural networks has initiated enormous progress in computational vision. A confluence of insights from biological vision, computational neuroscience, and statistical learning theory gave rise to deep convolutional networks, a model family that is now the face of the AI revolution and the *de facto* model of visual processing. Deep convolutional networks (DCNs) sit behind some of this century’s most incredible feats of visual intelligence. They can match, and sometimes surpass, human-level performance in recognizing over a 1000 object categories [1–4]. They can accurately detect faces, even amid distractors, and work with face forensic experts to improve their performance [5, 6]. They can reliably detect human actions in video clips [7, 8], support the synthesis of photo-realistic images [9], and are the engines behind the artificial agents that demonstrated mastery in long-term planning and decision-making [10–13]. The impact of the visual processing speed and ability of DCNs has already permeated scientific domains outside of computational vision. Testament to this is some of the applications we have explored in the cognitive and behavioral neurosciences [14–16].

DCNs’ utility for visual neuroscience stems from their ability to provide computational-level hypotheses about the nature of visual representations and the mechanisms that support their transformations along the cortical hierarchy. The fact that they can be “optimized” for specific training objectives meant that we can now assay through behaviorally-consistent models (for that objective) to find neurally-consistent ones. For instance, there is a consensus that the functional goal of the ventral visual pathway is to perform core object recognition [17]. Hence, towards identifying the “perfect” ventral stream model, DCNs performant on visual object recognition can be taxonomized per how well their internal representations predicted neural activity patterns from the ventral stream in primates performing the same task [18, 19]. Alternatively, one can iterate through training objectives (or hypothesized functional goals) for DCNs to determine the constraints under which neural consistency between models and primates emerges. For instance, it came to light that only training DCNs to predict self-motion parameters, as opposed to visual action recognition, yielded neural consistency with dorsal stream areas MT and MST [20]. Perhaps this approach can reveal novel aspects and functional roles of visual areas that we did not consider before.

Our primary critique of both these philosophies is that the behavioral and neural consistency achieved by DCNs is a by-product of their universal function approximation capability and not their mechanistic consistency with biological systems. This is relevant to address for two reasons. First, mechanistically consistent models of the visual system would also exhibit prowess in general visual cognition that extends beyond visual “recognition”. The circuits of the visual system support relational reasoning, spatial reasoning, visual working memory, mental imagery, and many other complex computations that are currently beyond the scope of DCNs. Second, the mechanisms of visual processing in biological networks make them robust. The lack of robustness to noise [21, 22] and novel statistical patterns [23, 24], is, in fact, one of the primary weaknesses of DCNs. This phenomenon, termed generalization, is a fundamental ingredient of visual intelligence. Despite the expansive visual diet, or exposure, our visual system receives from developmental stages through to adult life, we still commonly encounter non-canonical visual inputs that we have not seen before. Nevertheless, our behavioral responses in these situations are apt. We hypothesize that this bifurcation in behavior stems from biological systems’ ability to construct compositions of primitive pieces of knowledge dynamically.

Speaking of the mechanistic consistency of DCNs, one glaring omission is feedback. Vanilla DCNs are pre-attentive models of vision that excel at approximating bottom-up processes but do not have the scaffolding to express top-down and lateral feedback computations. Feedback is a dominant feature of cortical connectivity [25] and is known to play a critical role in shaping object-based attention, perceptual learning, visual working memory, and several other important functions [26, 27]. Its lack thereof in DCNs lies at the heart of the abovementioned shortcomings. Based on the universal function approximation theorem, one can argue that DCNs with sufficient network width and depth can approximate feedback computations. While this is true, it comes with the caveat of also requiring *infinite* data, thus making it impractical.

Deep convolutional recurrent networks (RNNs henceforth), a natural choice for feedback models, held immense promise for alleviating the bottlenecks that DCNs endure [28–30]. In fact, throughout this thesis, we have repeatedly relied upon the horizontal gated recurrent unit (hGRU), an RNN model family inspired by the anatomy of the visual cortex [31]. In prior work, the hGRU and related models demonstrably outperformed DCNs on a suite of tasks *and* generalized better while only using a fraction of DCN’s free parameters and training samples [31–34]. However, despite compelling evidence from biological and artificial vision suggesting that robust perception depended on such recurrent processing, the RNN approach for vision struggled to gain the same traction as DCNs. Why was this the case?

The answer was simple. Coupled with the expansive expressivity of large-scale RNNs was chaos. Recurrent network dynamics over long timescales tended to devolve into non-informative regimes, either trivial or unbounded. Practically, this meant that RNN model parameters were sensitive to even small perturbations, rendering learning via gradient descent challenging. While innovative architectural solutions to the learning problem, such as gates [35, 36] and skip connections [3], offered hope, progress was ultimately limited by another bottleneck. The standard algorithm for

spatiotemporal credit assignment in RNNs, “backpropagation through time” [37], imposed an onerous memory demand. Specifically, the memory required for training RNNs scaled linearly with the processing steps since optimization involved propagating an error signal through the entire latent trajectory. This made it infeasible to develop RNNs at scale.

Ironically, decades of theoretical and computational neuroscience research focused on characterizing, quantifying, and controlling chaos in recurrent networks, albeit at much smaller scales [38–41]. A sub-branch of this line of research, attractor dynamics modeling, focused on understanding temporal stereotypy in recurrent activity and the network mechanisms that support it. Temporal stereotypy can manifest as stationary, cyclic, or even chaotic patterns, each of which had consequences for the *type* of functions the network could carry out. For example, networks that reached stationary activity patterns were thought of as models of memory [42], while networks with cyclic attractors accounted for the various rhythms in brain signal [43]. Overall, attractor dynamics inherited the rigorous mathematical backing of non-linear dynamical systems theory and emerged as a unifying framework for neuroscience. Within this framework, network dynamics was the fundamental currency of computation, and shaping the dynamics is tantamount to prescribing a function for the network.

In this thesis, we provide one of the first steps in integrating the attractor dynamics perspective from neuroscience with large-scale, modern RNNs, and seek the best of both worlds. We introduce a general-purpose learning algorithm, C-RBP, that stabilizes large-scale RNNs, by forming attractor states, *without* compromising on network expressivity. Guarantees about an RNN’s stability allowed us to bypass the standard credit assignment scheme and devise a memory-efficient way to update its parameters during learning. This is one of our biggest wins, as it finally enabled the testing of computational models of feedback at scale. We demonstrate that C-RBP can train parameter-efficient architectures that can systematically generalize to out-of-distribution datasets.

C-RBP was a necessary first step in more ways than one. Until now, it was difficult to computationally test hypotheses about the role of recurrent dynamics and feedback in implementing “visual routines” [44]. Visual routines are primitive operations that are the building blocks of complex visual functions. Learning relevant routines and how to sequence them is a fundamental challenge in computational vision that has yet to be addressed systematically. In this thesis, through parallel investigations, we get at this problem. We observe that C-RBP optimized feedback models can learn an incremental grouping routine despite not being supervised for it. Astonishingly, the time courses of this model’s dynamics resembled human decision-making processes and hold much promise for further investigation. The same goes for studying the mechanisms underlying the visual routine of simulation. We also show that our models can learn flood-filling routines that improve performance on large-scale computer vision challenges that high-capacity DCNs dominate.

One benefit of developing effective visual perception models is that we can then use them to study how vision interfaces with higher-level cognitive processes. In isolation, there are multiple models of cognitive processes such as working memory [45–47] and decision making [48, 49]. However, only a tiny percentage of them consider the upstream effects of visual processing computations (for example, [50, 51]). And, almost none of them operate at scale on complex visual stimuli. The paucity

of models quantifying the interactions between the feedback circuits in the visual cortex and circuits supporting executive function motivated us to move in that direction.

The study of subsystems in isolation is, in fact, fraught with risk and must be avoided if possible. Consider the case study presented in Chapter. 5. The conclusions drawn about the functional role of neural substrates in a circuit can change non-trivially based on the nature of inputs from outside the system under consideration. In this specific example, we saw how noisy visual representations of complex stimuli, as opposed to idealized representations, can lead to the same neural substrate (synaptic efficacy) being used very differently and challenging the current dogma in the process. Similarly, in Chapter. 4, we develop a computational pipeline to demonstrate that the credit allocation problem that deep reinforcement learning agents face in discovering optimal policies for decision-making, is substantially impacted by the quality and semantic content of visual representations provided to the decision making components. Taking a step further, we suggest a potential mechanism through which these two systems can interact with each other to shape these “optimal” visual representations.

This thesis lays the groundwork for several questions that will hopefully be asked in the future. In adopting attractor dynamics for large-scale RNNs, we had limited ourselves to the special case of stationary dynamics or fixed points. Extending C-RBP to more complex attractors, such as limit cycles, might pose novel technical challenges. Furthermore, some of the more interesting use cases for neuroscience and cognitive psychology lie in a domain that is input-driven, which involves time-varying sensory inputs. The reinforcement learning example in Chapter. 4 and the working memory example in Chapter. 5 illustrate this point vividly. We adopted a two-phased approach in these two examples and trained our sensory and cognitive systems in succession. An improved version of C-RBP that can handle time-varying external inputs can greatly improve the efficiency and elevate the insights gained in this process. In scientific endeavors, solving a problem often opens up two newer ones. With the benefits of being able to train and utilize feedback models at scale comes questions about the interpretability of the network dynamics. Deriving analytical metrics to estimate properties of a trained model, such as reaction times and dynamical motifs (cf [52]), will play a pivotal role in the future.

6.1 References

- [1] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.
- [2] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).
- [3] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.
- [5] Ira Kemelmacher-Shlizerman et al. “The megaface benchmark: 1 million faces for recognition at scale”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4873–4882.
- [6] National Research Council et al. *Strengthening forensic science in the United States: a path forward*. National Academies Press, 2009.
- [7] Karen Simonyan and Andrew Zisserman. “Two-stream convolutional networks for action recognition in videos”. In: *Advances in neural information processing systems* 27 (2014).
- [8] Joao Carreira and Andrew Zisserman. “Quo vadis, action recognition? a new model and the kinetics dataset”. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.
- [9] Ian Goodfellow et al. “Generative adversarial networks”. In: *Communications of the ACM* 63.11 (2020), pp. 139–144.
- [10] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. en. In: *Nature* 518.7540 (Feb. 2015), pp. 529–533.
- [11] David Silver et al. “Mastering the game of Go without human knowledge”. en. In: *Nature* 550.7676 (Oct. 2017), pp. 354–359.
- [12] David Silver et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”. en. In: *Science* 362.6419 (Dec. 2018), pp. 1140–1144.
- [13] Oriol Vinyals et al. “Grandmaster level in StarCraft II using multi-agent reinforcement learning”. en. In: *Nature* 575.7782 (Nov. 2019), pp. 350–354.
- [14] Kelsey N Schuch* et al. “Discriminating between sleep and exercise-induced fatigue using computer vision and behavioral genetics”. In: *Journal of Neurogenetics* (2020), pp. 1–13.
- [15] Dima Amso et al. “Using Computational Analysis of Behavior To Discover Developmental Change In Memory-Guided Attention Mechanisms In Childhood”. In: (2021).
- [16] Lakshmi Narasimhan Govindarajan et al. “Neural Computing on a Raspberry Pi: Applications to Zebrafish Behavior Monitoring”. In: *24th International Conference on Pattern Recognition (Workshop)*. 2018.

- [17] Mortimer Mishkin, Leslie G Ungerleider, and Kathleen A Macko. “Object vision and spatial vision: two cortical pathways”. In: *Trends in neurosciences* 6 (1983), pp. 414–417.
- [18] Daniel LK Yamins et al. “Performance-optimized hierarchical models predict neural responses in higher visual cortex”. In: *Proceedings of the national academy of sciences* 111.23 (2014), pp. 8619–8624.
- [19] Martin Schrimpf et al. “Brain-score: Which artificial neural network for object recognition is most brain-like?” In: *BioRxiv* (2020), p. 407007.
- [20] Patrick Mineault et al. “Your head is there to move you around: Goal-driven models of the primate dorsal pathway”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 28757–28771.
- [21] Sandy Huang et al. “Adversarial attacks on neural network policies”. In: *arXiv preprint arXiv:1702.02284* (2017).
- [22] Aleksander Madry et al. “Towards deep learning models resistant to adversarial attacks”. In: *arXiv preprint arXiv:1706.06083* (2017).
- [23] Drew Linsley et al. “Robust neural circuit reconstruction from serial electron microscopy with convolutional recurrent networks”. In: (Nov. 2018). arXiv: 1811.11356 [cs.CV].
- [24] Zheyang Shen et al. “Towards out-of-distribution generalization: A survey”. In: *arXiv preprint arXiv:2108.13624* (2021).
- [25] Nikola T Markov et al. “Anatomy of hierarchy: feedforward and feedback pathways in macaque visual cortex”. In: *Journal of Comparative Neurology* 522.1 (2014), pp. 225–259.
- [26] Charles D Gilbert and Wu Li. “Top-down influences on visual processing”. In: *Nature Reviews Neuroscience* 14.5 (2013), pp. 350–363.
- [27] Lothar Spillmann, Birgitta Dresch-Langley, and Chia-Huei Tseng. “Beyond the classical receptive field: The effect of contextual stimuli”. In: *Journal of Vision* 15.9 (2015), pp. 7–7.
- [28] Qianli Liao and Tomaso Poggio. “Bridging the gaps between residual learning, recurrent neural networks and visual cortex”. In: *arXiv preprint arXiv:1604.03640* (2016).
- [29] Kohitij Kar et al. “Evidence that recurrent circuits are critical to the ventral stream’s execution of core object recognition behavior”. en. In: *Nat. Neurosci.* (Apr. 2019).
- [30] Kohitij Kar et al. “Evidence that recurrent circuits are critical to the ventral stream’s execution of core object recognition behavior”. en. In: *Nat. Neurosci.* 22.6 (June 2019), pp. 974–983.
- [31] Drew Linsley et al. “Learning long-range spatial dependencies with horizontal gated recurrent units”. In: *Advances in neural information processing systems* 31 (2018).
- [32] Junkyung Kim et al. “Disentangling neural mechanisms for perceptual grouping”. In: *arXiv preprint arXiv:1906.01558* (2019).
- [33] Drew Linsley et al. “Tracking Without Re-recognition in Humans and Machines”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 19473–19486.

- [34] Drew Linsley et al. “Recurrent neural circuits for contour detection”. In: *International Conference on Learning Representations* (2020).
- [35] S Hochreiter and J Schmidhuber. “Long short-term memory”. en. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780.
- [36] Junyoung Chung et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *arXiv preprint arXiv:1412.3555* (2014).
- [37] P J Werbos. “Backpropagation through time: what it does and how to do it”. In: *Proc. IEEE* 78.10 (Oct. 1990), pp. 1550–1560.
- [38] Haim Sompolinsky and Ido Kanter. “Temporal association in asymmetric neural networks”. In: *Physical review letters* 57.22 (1986), p. 2861.
- [39] Haim Sompolinsky, Andrea Crisanti, and Hans-Jurgen Sommers. “Chaos in random neural networks”. In: *Physical review letters* 61.3 (1988), p. 259.
- [40] Carl Van Vreeswijk and Haim Sompolinsky. “Chaos in neuronal networks with balanced excitatory and inhibitory activity”. In: *Science* 274.5293 (1996), pp. 1724–1726.
- [41] Steven H Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.
- [42] John J Hopfield. “Neural networks and physical systems with emergent collective computational abilities.” In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.
- [43] Mikail Khona and Ila R Fiete. “Attractor and integrator networks in the brain”. In: *arXiv preprint arXiv:2112.03978* (2021).
- [44] Shimon Ullman. “Visual routines”. In: *Readings in computer vision*. Elsevier, 1987, pp. 298–328.
- [45] G. Mongillo, O. Barak, and M. Tsodyks. “Synaptic theory of working memory”. In: *Science* 319.5869 (Mar. 2008), pp. 1543–1546.
- [46] N. Y. Masse et al. “Circuit mechanisms for the maintenance and manipulation of information in working memory”. In: *Nat. Neurosci.* 22.7 (July 2019), pp. 1159–1167.
- [47] Mikael Lundqvist, Pawel Herman, and Anders Lansner. “Theta and gamma power increases and alpha/beta power decreases with memory load in an attractor network model”. In: *Journal of cognitive neuroscience* 23.10 (2011), pp. 3008–3020.
- [48] Megan Wang, Daniel Arteaga, and Biyu J He. “Brain mechanisms for simple perception and bistable perception”. In: *Proceedings of the National Academy of Sciences* 110.35 (2013), E3350–E3359.
- [49] Xiao-Jing Wang. “Probabilistic decision making by slow reverberation in cortical circuits”. In: *Neuron* 36.5 (2002), pp. 955–968.
- [50] F. Bouchacourt and T. J. Buschman. “A Flexible Model of Working Memory”. In: *Neuron* 103.1 (July 2019), pp. 147–160.

- [51] Thomas E Hazy, Michael J Frank, and Randall C O'reilly. "Towards an executive without a homunculus: computational models of the prefrontal cortex/basal ganglia system". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 362.1485 (2007), pp. 1601–1613.
- [52] Laura Driscoll, Krishna Shenoy, and David Sussillo. "Flexible multitask computation in recurrent networks utilizes shared dynamical motifs". In: *bioRxiv* (2022). DOI: 10.1101/2022.08.15.503870. eprint: <https://www.biorxiv.org/content/early/2022/08/15/2022.08.15.503870.full.pdf>. URL: <https://www.biorxiv.org/content/early/2022/08/15/2022.08.15.503870>.