Computational Models of Learning and Representations in the Hippocampal Formation

Tom M. George

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Sainsbury Wellcome Centre for Neural Circuits and Behaviour

University College London

June 26, 2025

...In that Empire, the Art of Cartography attained such Perfection that the map of a single Province occupied the entirety of a City, and the map of the Empire, the entirety of a Province. In time, those Unconscionable Maps no longer satisfied, and the Cartographers Guilds struck a Map of the Empire whose size was that of the Empire, and which coincided point for point with it. The following Generations, who were not so fond of the Study of Cartography as their Forebears had been, saw that that vast Map was Useless, and not without some Pitilessness was it, that they delivered it up to the Inclemencies of Sun and Winters. In the Deserts of the West, still today, there are Tattered Ruins of that Map, inhabited by Animals and by Beggars; in all the Land there is no other Relic of the Disciplines of Geography.

— *On Exactitude in Science*, Jorge Luis Borges

I, Tom M. George, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

This thesis explores how neural systems learn and use internal representations to support flexible behaviour, focusing on the mammalian spatial memory system and cognitive map. It introduces new computational tools and biologically plausible models that link neural structure and dynamics to spatial cognition.

Foundational to this work is RatInABox, an open-source toolkit now widely used for simulating realistic navigation and hippocampal activity. This platform enables the rapid prototyping of models that jointly capture behavioural trajectories and neural representations, including place and grid cells.

The first model leverages this toolkit to demonstrate a biologically plausible mechanism for learning predictive representations via spike-timing dependent plasticity and theta phase precession. This mechanism bridges the timescale gap between behaviour and synaptic plasticity to enable fast and flexible learning.

A second study introduces a generative model of the hippocampal-entorhinal loop that unifies path integration and mental simulation. It shows how local Hebbian learning, scheduled by theta oscillations, can give rise to ring attractor dynamics within a normative Helmholtz machine framework.

The final contribution is SIMPL, an efficient algorithm for latent variable discovery from high-dimensional neural data. By recursively fitting latent trajectories and tuning curves, SIMPL achieves state-of-the-art performance while enhancing the interpretability and precision of neural representations.

Collectively, these contributions advance our understanding of how biological systems learn and represent the world, and provide new models and tools for research at the intersection of neuroscience and artificial intelligence.

Impact Statement

The speculative benefits inside academia might be to the discipline and future of scholarship, research methods or methodology, the curriculum; they might be within neuroscience and potentially within other research areas.

The speculative benefits outside academia might be to commercial activity, social enterprise, professional practice, clinical use, public health, public policy design, public service delivery, laws, public discourse, culture, the quality of the environment or quality of life.

There might not even be any benefits at all.

Publications Arising

This thesis details work covered in five peer-reviewed publications as follows:

- **Chapter 1**: Tom M. George, Mehul Rastogi, William de Cothi, Claudia Clopath, Kimberly L. Stachenfeld, and Caswell Barry (2024). "RatInABox, a toolkit for modelling locomotion and neuronal activity in continuous environments". In: *eLife*. DOI: 10.7554/elife.85274
- **Chapter 2**: Tom M. George (2023). "Theta sequences as eligibility traces: A biological solution to credit assignment". In: *International Conference on Learning Representations 2023 (TinyPapers track)*. DOI: 10.48550/arXiv.2305.08124
- **Chapter 3**: Tom M. George, William de Cothi, Kimberly L. Stachenfeld, and Caswell Barry (Mar. 2023a). "Rapid learning of predictive maps with STDP and theta phase precession". In: *eLife* 12. DOI: 10.7554/elife.80663
- **Chapter 4**: Tom M. George, Kimberly L. Stachenfeld, Caswell Barry, Claudia Clopath, and Tomoki Fukai (2023b). "A generative model of the hippocampal formation trained with theta driven local learning rules". In: *Thirty-seventh Conference on Neural Information Processing Systems*. DOI: 10.1101/2023.12.12.571268
- **Chapter 5**: Tom M. George, Pierre Glaser, Kimberly L. Stachenfeld, Caswell Barry, and Claudia Clopath (2025). "SIMPL: Scalable and hassle-free optimisation of neural representations from behaviour". In: *The Thirteenth International Conference on Learning Representations*. DOI: 10.1101/2024.11.11.623030

UCL Research Paper Declaration Forms can be found in Appendix F.

Acknowledgements

For this thesis I would like to thank my mentors—and there have been many—for shining a light on my path through academia and giving me the confidence to follow it.
I would like to thank my friends, in the UK and across the world, for taking me to the pub, bar or izakaya when I needed it, and making these years so enjoyable.
And I would like to thank my family, especially my Dad, who have always believed in me and encouraged me from the very beginning.

More than anything, I would like to thank Lili. Her unwavering support over the last nine years has meant more to me than she will ever know.

Contents

List of Figures

List of Tables

Chapter 0

Introduction

Intelligent agents, both biological and artificial, must learn to represent the external world and themselves within it to support flexible behaviours, such as navigation, planning, and decision-making. This thesis explores how neural systems achieve these goals by learning and utilising *internal representations*. My research addresses three fundamental questions: how neural systems learn, what representations neural systems use, and how neural dynamics and representations interact to enable flexible behaviour. By developing computational models and a toolkit for constructing them, this work aims to help reconcile biological observations with statistical and machine learning frameworks for intelligent systems, providing novel interpretations for both.

0.1 The Mammalian Spatial Memory System: A Foundational Model Domain

The mammalian spatial memory system, particularly the hippocampal formation, serves as the primary model domain for this thesis. The hippocampus, a small *c*-shaped structure in the medial temporal lobe and the most studied of all brain regions, is known to play a critical role in memory (Scoville et al. 1957) and spatial navigation (Tolman et al. 1930; O'Keefe et al. 1978; Morris et al. 1982). Decades of research have illuminated a rich landscape of learning phenomena (Bi et al. 1998; Carr et al. 2011; Bittner et al. 2017), architectures (Bush et al. 2014), representations (Moser et al. 2017), and dynamics (Carr

et al. 2011; Skaggs et al. 1996a; Buhry et al. 2011; Sanders et al. 2015) within the hippocampal formation, making it an ideal testbed for understanding how neural systems learn and represent information. The hippocampal formation (here taken to include the entorhinal cortex, subiculum, and dentate gyrus) is known to represent self-location through the activity of a rich diversity of functional cell types including "place cells," "grid cells," and "boundary vector cells" (Moser et al. 2008). Understanding how these neural components interact to support sophisticated cognitive functions is a central focus of my research.

Despite extensive research, much about the hippocampal formation remains unknown. By some measures, our high-level computational and algorithmic understanding—for example, orthogonalising sensory inputs (Treves et al. 1994), memory consolidation (Scoville et al. 1957), encoding self-location (O'Keefe et al. 1978), and path integration via the grid code (McNaughton et al. 2006; Fuhs et al. 2006)—has outpaced our knowledge of the biological mechanisms enabling these functions. This thesis directly tackles this problem by investigating how successful theoretical models might *actually* be implemented within the brain, bridging the gap.

0.1.1 Spatial Navigation and Spatial Cognition

Moving around the world—finding food, avoiding predators, and navigating to ecologically relevant locations—is a fundamental task for almost all animals and a distinguishing feature of intelligent life. For nearly a century, available evidence has suggested that animals build *internal cognitive maps*: systems of neurons which describe (or "encode") an animal's environment and their location within it. This idea was first proposed by Tolman et al. (1930) who noticed that rats with prior experience of a maze could navigate to a subsequent goal location faster than animals without. This suggests that navigation is not merely a simple stimulus-response behaviour but involves learning a temporally-sustained map of the world. Support for this hypothesis was further solidified in 1948 (Tolman 1948) with the discovery that experienced rats could navigate to the goal through nearly-optimal routes even when the previously learned

path was blocked.

Whereas spatial *navigation* refers to the process of locating and locomoting oneself through the environment, spatial *cognition* refers to a broader set of cognitive functions that involve the representation and manipulation of spatial information. Examples include remembering past locations (Wilson et al. 1994; Nadel et al. 1997), planning future routes (Spiers et al. 2006; Pfeiffer et al. 2013), understanding spatial relationships between objects (Høydal et al. 2019), integrating vestibular cues (Castillo et al. 1954), and simulating trajectories (so-called "mind-travel" (Sanders et al. 2015)). These functions, as we will see, place substantial constraints on how and what the mammalian spatial memory system must learn.

0.1.2 Hippocampal representations form the basis of the cognitive maps

The theoretical construct of the cognitive map found its first biological grounding with the discovery of "place cells" in the hippocampus by O'Keefe et al. (1971). These neurons exhibit location-specific firing, becoming active only when an animal enters a particular region of its environment (the cell's "place field"), providing direct evidence that the hippocampus encodes an animal's location. Their discovery led to the influential theory that the hippocampus is the key locus of the cognitive map.

Over three decades later, the discovery of "grid cells" in the medial entorhinal cortex (mEC)—a primary input to the hippocampus—by Hafting et al. (2005) added a crucial new dimension. Grid cells fire at multiple locations, with their firing fields forming a periodic triangular lattice that tiles the entire environment. This regular structure is thought to provide something closer to a metric or a coordinate system for the spatial memory system—often likened to the gridlines on a map—enabling path integration and measurement of distances and vectors (McNaughton et al. 2006; Burak et al. 2009). Along with place cells, their discovery kick-started a new wave of theoretical research into what types of neural representations are optimal for spatial cognition and

earned their discoverers a Nobel Prize.

While place and grid cells are the foundational building blocks of the spatial map, a map designed for flexible navigation must encode more than just an agent's current location. It must also capture the relationships *between* places and their value for goal-directed behaviour. The successor representation (SR), a particular focus of Chapter 3, is a powerful theoretical framework that addresses this need by positing that the hippocampus encodes not only the animal's current position but an expectation over its future positions (Dayan 1993; Stachenfeld et al. 2017). By bridging the principles of reinforcement learning with the firing properties of hippocampal neurons, the SR framework explains how these representations support not only localisation but also flexible planning and decision-making.

The hippocampal formation is not limited to place and grid cells. A diverse array of other cell types have been identified, including boundary vector cells (O'Keefe et al. 1996; Lever et al. 2009), object vector cells (Høydal et al. 2019), head direction cells (Taube et al. 1990), speed cells (Kropff et al. 2015), time cells (Pastalkova et al. 2008), and more. These additional representations contribute to a rich tapestry of spatial coding mechanisms that support navigation and cognition. Whilst many theoretical models have been developed to prospectively (O'Keefe et al. 1996) or retrospectively (Burgess et al. 2007) explain the firing properties of these cells, the challenge remains to unify these findings and understand how such diverse representations interact to form a coherent cognitive map.

0.1.3 The role of learning and dynamics in hippocampal function

The representations that form the cognitive map are not static; they are shaped by experience and are constantly updated through the interplay of synaptic plasticity and network dynamics. Learning in the hippocampus is classically attributed to Hebbian-like mechanisms, such as long-term potentiation (LTP) (Bliss et al. 1973), which strengthen the connections between neurons that

fire together (Hebb 1949), as well as STDP, which adjusts synaptic weights based on the relative timing of pre- and post-synaptic spikes (Markram et al. 1997; Bi et al. 1998). This allows for the rapid formation of place fields as an animal explores a new environment (Bittner et al. 2017) and the association of locations with salient events or rewards (Hollup et al. 2001).

Furthermore, hippocampal function is critically dependent on internally generated neural dynamics, most notably theta oscillations (Green et al. 1954) and sharp-wave ripples (SWRs) (Buzsáki et al. 1992). During active exploration, the hippocampal local field potential (LFP) exhibits a prominent 4-12 Hz theta oscillation, which is thought to coordinate neural activity to encode ongoing experiences in real-time and is studied extensively in Chapter 4. The associated phenomenon of "theta phase precession" (O'Keefe et al. 1993), where place cells fire at progressively earlier phases of the theta cycle as an animal traverses a place field, suggests a mechanism for encoding temporal sequences of events within a single oscillatory cycle (Skaggs et al. 1996b) and is a focal mechanism in Chapters 2 and 3.

In contrast, during periods of rest or quiescence, the hippocampus is dominated by SWRs—brief, high-frequency bursts of activity. During SWRs, the hippocampus "replays" sequences of place cell activity corresponding to past (Wilson et al. 1994) or potential future (Pfeiffer et al. 2013) trajectories, but on a heavily compressed timescale (Nádasdy et al. 1999). This replay is thought to be crucial for memory consolidation and the transfer of information from the hippocampus to neocortical regions (Marr 1971; Buzsáki 1989), allowing for the integration of new experiences into long-term memory.

Of particular importance to this thesis is the fact that neural dynamics and neural representations are rarely independent concepts. As we will see in Chapter 5, neural dynamics can modify spiking in such a way that, unaccounted for, blurs or distorts observed tuning curves. This has important implications for how we interpret neural data and understand the underlying cognitive processes. Furthermore, in Chapters 2 to 4, we will see how neural dynamics

are not just critical for how spatial cognition is performed *after* learning, but are also critical for learning itself.

0.1.4 Theoretical and Computational Models of Hippocampal Function

To bridge the gap between biological observation and functional understanding, the field—along with this thesis—relies heavily on theoretical and computational models. This reliance operates under the principle, articulated by statistician George Box, that "all models are wrong, but some are useful". These models serve as formal hypotheses for how the anatomical structures, neural representations, and network dynamics of the hippocampus give rise to its cognitive functions while also acting as engines for generating new, testable predictions (Epstein 2008). The models exist across multiple levels of abstraction, each offering a unique trade-off between biological realism and explanatory power.

At one end of the spectrum are **biophysically detailed models**, which aim to simulate the behaviour of individual neurons and even synapses with a high degree of fidelity, often incorporating specific ion channels (Hodgkin et al. 1952; Chen et al. 2022) and cellular morphologies (Ascoli et al. 2007). While these models are invaluable for understanding how specific cellular mechanisms contribute to network phenomena (e.g., the generation of theta rhythms (Buzsáki 2002)), their complexity makes it difficult to extract general computational principles, and, to date, they have never displayed truly "intelligent" behaviours.

At a higher level of abstraction are **connectionist and network-level models**. These models, such as continuous attractor networks (Zhang 1996), simplify the behaviour of individual neurons but focus on the collective dynamics of the network (Hopfield 1982). Attractor models have been particularly influential (Wills et al. 2005) in the hippocampal and spatial modelling literature, proposing that place cell activity emerges from recurrent connectivity within the hippocampus (Rolls et al. 2006). In this view, the network can "settle" into

a stable state or "attractor" that corresponds to a specific location, providing a robust mechanism for self-localisation and memory completion.

Finally, **normative, or functional-level, models** operate at the highest level of abstraction, often drawing inspiration from statistics (Knill et al. 2004), machine learning (Banino et al. 2018; Hassabis et al. 2017), and reinforcement learning (Schultz et al. 1997). These models prioritise the question of *what* the hippocampus computes over *how* it is implemented biologically. Examples include Bayesian models that treat hippocampal activity as encoding a probability distribution over the animal's location (Deneve et al. 2001), the aforementioned successor representation (SR), which frames hippocampal function in terms of predicting future states (Stachenfeld et al. 2017), as well as the Tolman-Eichenbaum Machine (TEM) (Whittington et al. 2020), which posits that the hippocampus binds sensory inputs with internally generated predictions to support flexible navigation and planning. Notably, models like TEM are explicitly hybrid, incorporating both high-level normative ideas and concrete network-level mechanisms, thereby representing an important step towards reconciling these different levels of analysis.

While all these modelling approaches have yielded critical insights, a significant challenge remains in reconciling the elegance and power of normative theories with the messy, constrained reality of biological hardware (Marr 1982; Barak 2017). This thesis contributes directly to this effort by developing and analysing models that are inspired by normative principles but are explicitly designed to respect the known anatomical and physiological constraints of the hippocampal formation. I believe this exercise is worthwhile for two principal reasons: Firstly, grounding abstract theories in biology provides a crucial existence proof, demonstrating that the computations they propose are actually achievable within the known constraints of neural hardware. Secondly, by linking normative functions to specific biological mechanisms, our theories often become more generalisable (Carandini et al. 2011). A principle discovered in the hippocampus can then illuminate our understanding of other brain regions

that employ similar circuits or synaptic learning rules, helping to build a more unified picture of neural computation.

0.1.5 Theoretical and Computational Model*ling* of Hippocampal Function

The modern toolkit for computational neuroscience is largely built on the Python programming language, benefiting from its ease of use and extensive ecosystem of open-source libraries for scientific computing. Despite this common foundation, a significant challenge persists in how models are developed and shared. The prevailing culture is often one of bespoke creation, where individual labs or researchers build unique, single-use codebases for their specific questions. While this approach provides a great deal of flexibility, it creates a fragmented landscape that is inefficient and possibly even detrimental to overall progress. This fragmentation leads to a massive duplication of effort as researchers repeatedly reinvent foundational components—for example, coding up a simple motion model, or the receptive field of a grid cell—and more importantly, it erects barriers to reproducibility and direct model comparison.

Fortunately, this challenge has not gone unrecognised, and in recent years the field has made large strides towards open, collaborative science. Landmark open-source endeavours are transforming how we analyse and share empirical data. Toolkits for animal pose estimation like DeepLabCut (Mathis et al. 2018) and SLEAP (Pereira et al. 2022) have standardised complex behavioural analysis, while large-scale data repositories from the Allen Institute for Brain Science (Vries et al. 2023) and standards like Neurodata Without Borders (NWB) (Rübel et al. 2022) have democratised access to neural recordings. Similar movements towards standardisation have also been emerging in the domain of in-silico experimentation, particularly for low-level spiking neural simulators (Hines et al. 1997; Goodman 2008). Despite this, the community still lacks widely adopted, high-level toolkits specifically for generating synthetic data or for rapidly prototyping, training, and comparing complex hippocampal models, a topic we directly tackle in Chapter 1.

While the models themselves often receive more attention, the shared toolkits used to build them are as fundamental to scientific progress. Creating robust, reusable software is a critical part of our responsibility as scientists because it directly supports the core principles of reproducibility and collaboration. This infrastructure work is a necessary investment in the long-term health and efficiency of our field.

0.2 Overarching Themes and Methods

The preceding sections have established a central challenge in neuroscience: reconciling high-level theories of cognition with the complex, constrained realities of biological circuits. This thesis confronts this challenge directly through research that is loosely organised around two key themes.

The first is the **development and analysis of computational models that are explicitly designed to be biologically plausible.** My research strategy is to use these models as a bridge between the functional principles of machine learning and the known anatomy, physiology, and dynamics of the hippocampal formation. The second theme is the **development and validation of computational tools to enable better computational model building and data visualisation.**

0.3 Key Contributions of This Thesis

This thesis presents four primary research contributions, each targeting a distinct gap between high-level theory and biological mechanism and aligned with the themes outlined above.

0.3.1 A Standardised Toolkit for Reproducible Hippocampal Modelling and Data Generation

A significant practical barrier in computational neuroscience is the lack of standardised environments for building and testing models, leading to fragmented and difficult-to-replicate research. To address this, this thesis introduces the open-source Python package, *RatInABox*, which has seen

significant adoption by the community—with over 55,000 downloads to date—and provides a standardised platform for simulating rodent locomotion and the activity of spatially modulated neurons. It is designed to facilitate the rapid prototyping and testing of hypotheses about how behaviour, representations, and learning interact in the context of the cognitive map. By providing a common framework for simulation, this package aims to reduce fragmentation and promote more efficient and reproducible research practices (Barnes 2010; Wilson et al. 2017). This work is a direct contribution to the second theme of this thesis: the development of computational tools to enable better model building.

0.3.2 A Biologically Plausible Mechanism for Learning Predictive Maps

A leading normative theory is that the hippocampus builds a "predictive map" of the environment, formalised by the successor representation (SR) (Dayan 1993). This framework powerfully explains how animals can navigate flexibly by representing not just their current location, but a predictive landscape of future locations (Stachenfeld et al. 2017). A major unresolved issue, however, is *how* such a predictive map could be learned by hippocampal circuits. The canonical algorithm for learning SRs, temporal difference (TD) learning, is difficult to map onto known hippocampal biology, particularly given the very different timescales of spike-timing based synaptic plasticity ($\mathcal{O}(20\text{ ms})$) (Markram et al. 1997; Bi et al. 1998) and behaviour ($\mathcal{O}(10\text{ s})$).

This thesis proposes and validates a novel, biologically plausible mechanism for learning the SR. Specifically, I show how theta phase precession allows STDP, a learning rule sensitive to millisecond-timescale spike timing, to rapidly integrate information over behavioural timescales of many seconds. The resulting model learns synaptic weights that closely approximate the SR, successfully explaining empirical observations like the goal-directed skewing of place fields (Mehta et al. 1997). In line with the first theme of this thesis, this work forges a deep, previously unappreciated theoretical link between the

cellular mechanism of STDP and the algorithmic principle of TD learning, providing a more concrete mechanism for how predictive maps can be learned in the brain.

0.3.3 A Generative Model for Path Integration and Mental Simulation

Beyond representing known locations, the cognitive map supports generative functions, such as estimating one's position by integrating self-motion cues (path integration) (McNaughton et al. 2006) or simulating future trajectories ("mind travel") (Johnson et al. 2007; Buckner et al. 2007). A promising class of models, including the Tolman-Eichenbaum Machine (Whittington et al. 2020), posits that the brain accomplishes this by operating as a predictor-comparator circuit, constantly matching internally generated predictions against incoming sensory evidence (Rao et al. 1999). However, these models often rely on learning algorithms like backpropagation-through-time or make unrealistic assumptions about pre-existing neural connectivity (Lillicrap et al. 2020), leaving their biological feasibility an open question.

To address this, this thesis introduces the "*Helmholtz Hippocampus*," a generative model of the hippocampal-entorhinal loop that performs these functions using only local, Hebbian-like learning rules. The central hypothesis is that theta oscillations act as a control signal, rapidly switching multi-compartmental neurons between two distinct phases of operation—one for encoding bottom-up sensory input and one for generating top-down internal predictions. This mechanism, analogous to the wake-sleep algorithm of a Helmholtz Machine (Hinton et al. 1995; Dayan et al. 1995), allows the network to self-organise a self-sustaining continuous ring attractor structure from unstructured sensory input (Zhang 1996). The neurons in this emergent attractor network share numerous properties with entorhinal grid cells and provide a robust substrate for path integration. This work, which aligns with the first theme of this thesis, demonstrates how fundamental biophysical properties like neural oscillations can be harnessed to implement sophisticated generative

computations.

0.3.4 A Practical Method for Discovering Latent Neural Representations

A foundational challenge in neuroscience is to accurately characterise the relationship between neural activity (spikes) and the variables it represents. Accurately doing so tells us a lot about the function of a system and can give clues to the underlying mechanisms or computational principles at play. Often, however, both the variables being encoded by a neural system and how they map to neural activity (their "tuning curves") are, a priori, unknown. This creates a "catch-22" scenario, well studied in both neuroscience and machine learning, known as the *latent variable problem* (Paninski et al. 2007). A majority of hippocampal research has historically bypassed this problem by assuming the latent variable is a directly measurable behavioural correlate, such as the animal's physical position. This assumption was instrumental in the development of the cognitive map theory and the discovery of functional cell types like place cells.

This reliance on observed behaviour is, however, fundamentally limiting, as an animal's internal latent can diverge from its externally measured state (Low et al. 2018)—a discrepancy made explicitly evident during dynamic phenomena like theta sequences and memory replay (Wilson et al. 1994). Such mismatches are a subtle but important roadblock in our goal to bridge high-level theories of neural function with low-level biological mechanisms, since doing so requires knowing the tuning curves of individual neurons with considerable precision. Traditional analyses, which presuppose a direct correspondence between firing and behaviour, can be distorted by this discrepancy, leading to an incomplete understanding of the neural code. This often results in tuning curves that appear noisy, weak, or spatially imprecise, masking the true fidelity of the underlying neural representation (Pillow et al. 2008).

To overcome this limitation, this thesis presents SIMPL *(Scalable Iterative Maximisation of Population-coded Latents)*, a novel and computationally

efficient method for latent variable discovery. SIMPL operates by recursively optimising both the neural tuning curves and the trajectory of the underlying latent variable, using the observed behaviour only as an initial "best guess." It thus synergises the interpretability and speed of traditional analysis with the statistical power of modern approaches to latent variable modelling, such as expectation-maximisation (Dempster et al. 1977). When applied to hippocampal recordings, SIMPL uncovers place fields that are sharper, more numerous, and more stable than those inferred from behaviour alone, suggesting that the brain's cognitive map may be significantly more precise than previously thought. As a contribution to the second theme of this thesis, SIMPL provides a more accurate and robust lens for investigating the link between neural dynamics and representation across diverse brain regions.

0.4 Thesis Structure

The remainder of this thesis is organised as follows:

- **Chapter 1: The RatInABox toolkit** details the open-source Python toolkit developed to facilitate realistic simulations of rodent locomotion and associated neural activity, serving as a foundational platform for all subsequent modelling work.

- **Chapter 2: Theta Sequences as Eligibility Traces** explores how theta sequences in the hippocampus can function analogously to eligibility traces in reinforcement learning, providing a biological solution to the long-term credit assignment problem. It serves as a theoretical precursor to the more plausible learning mechanisms explored in Chapter 3 and may be skipped by readers primarily interested in the latter.

- **Chapter 3: Learning Predictive Maps with STDP and Theta** presents a detailed model demonstrating that STDP, when augmented by theta phase precession, is sufficient to rapidly learn a close approximation of the successor representation, consistent with hippocampal data.

- **Chapter 4: The Hippocampal Generative Model** introduces a generative model of the hippocampal-entorhinal loop that performs path integration and mental simulation using local learning rules and oscillation-controlled message passing.
- **Chapter 5: SIMPL: A Neural Latent Variable Model** presents an efficient method for neural latent discovery that optimises tuning curves and latent trajectories from spiking data.
- **Chapter 6: General Conclusions** synthesises the findings across the thesis and discusses their broader implications as well as future directions for research in computational neuroscience.

0.5 Broader Impact and Future Directions

The research presented in this thesis collectively advances our understanding of learning and representation by providing concrete, biologically plausible models for abstract cognitive functions. It also contributes a new, widely-adopted software tool to the computational neuroscience community. A central insight emerging from this work is the critical and dynamic interplay between synaptic structure (the learned weights of the network) and ongoing neural dynamics (the latent factors that shape activity in real-time) (Buonomano et al. 2009).

Understanding when and why the brain relies on modifying long-term structure versus modulating short-term dynamics is a key question for future research. This distinction mirrors important debates in modern artificial intelligence, such as the difference between "in-weights" and "in-context" (Brown et al. 2020) learning in large language models, or the trade-offs between model-based and model-free reinforcement learning (Daw et al. 2005; Geerts et al. 2020), as well as in neuroscience, such as the distinction between episodic and working memory (El-Gaby et al. 2024; Whittington et al. 2025). The path forward lies in developing a more unified theory that explains how these two modes of computation are balanced and integrated across different

cognitive domains. The models and methods developed in this thesis provide a foundational step in that direction.

Chapter 1

RatInABox, a toolkit for modelling locomotion and neuronal activity in continuous environments

Summary

Generating synthetic locomotory and neural data is a useful yet cumbersome step commonly required to study theoretical models of the brain's role in spatial navigation. This process can be time consuming and, without a common framework, makes it difficult to reproduce or compare studies which each generate test data in different ways. In response, I present RatInABox, an open-source Python toolkit designed to model realistic rodent locomotion and generate synthetic neural data from spatially modulated cell types. This software provides users with (i) the ability to construct one- or two-dimensional environments with configurable barriers and visual cues, (ii) a physically realistic random motion model fitted to experimental data, (iii) rapid online calculation of neural data for many of the known self-location or velocity selective cell types in the hippocampal formation (including place cells, grid cells, boundary vector cells, head direction cells) and (iv) a framework for constructing custom

cell types, multi-layer network models and data- or policy-controlled motion trajectories. The motion and neural models are spatially and temporally continuous as well as topographically sensitive to boundary conditions and walls. It is demonstrated that out-of-the-box parameter settings replicate many aspects of rodent foraging behaviour such as velocity statistics and the tendency of rodents to over-explore walls. Numerous tutorial scripts are provided, including examples where RatInABox is used for decoding position from neural data or to solve a navigational reinforcement learning task. I hope this tool will significantly streamline computational research into the brain's role in navigation.

1.1 Introduction: The Need for Standardized Tools in Computational Neuroscience

Computational modelling provides a means to understand how neural circuits represent the world and influence behaviour, interfacing between experiment and theory to express and test how information is processed in the brain. Such models have been central to understanding a range of neural mechanisms, from action potentials (Hodgkin et al. 1952) and synaptic transmission between neurons (Castillo et al. 1954), to how neurons represent space and guide complex behaviour (Hartley et al. 2000; Hartley et al. 2004; Byrne et al. 2007; Banino et al. 2018; Cothi et al. 2022a). Relative to empirical approaches, models can offer considerable advantages, providing a means to generate large amounts of data quickly with limited physical resources, and are a precise means to test and communicate complex hypotheses. To fully realise these benefits, computational modelling must be accessible and standardised, something which has not always been the case.

Spurred on by the proposition of a "cognitive map" (Tolman et al. 1930), and the discovery of neurons with position- (O'Keefe et al. 1971), velocity- (Sargolini et al. 2006; Kropff et al. 2015) and head direction- (Taube et al. 1990)selective receptive fields in the hippocampal formation, understanding

the brain's role in navigation and spatial memory has been a key goal of the neuroscience, cognitive science, and psychology communities. In this field it is common for theoretical or computational models to rely on artificially generated data sets. For example, for the direct testing of a normative model, or to feed a learning algorithm with training data from a motion model used to generate a time series of states, or feature-vectors. Not only is this data more cost-effective, quicker to acquire, and less resource-intensive than conducting spatial experiments (no rats required), but it also offers the advantage of being flexibly hand-designed to support the validation or refutation of theoretical propositions. Indeed, many past (Mehta et al. 2000; Burak et al. 2009; Gustafson et al. 2011) and recent (Stachenfeld et al. 2017; Cothi et al. 2020a; Bono et al. 2021; George et al. 2023a; Banino et al. 2018; Schaeffer et al. 2022; Benna et al. 2021) models have relied on artificially generated movement trajectories and neural data.

Artificially generating data can still be a bottleneck in the scientific process. I observe a number of issues: First, the lack of a universal standard for trajectory and cell activity modelling hinders apples-to-apples comparisons between theoretical models whose conclusions may differ depending on the specifics of the models being used. Secondly, researchers must begin each project reinventing the wheel, writing software capable of generating pseudo-realistic trajectories and neural data before the more interesting theoretical work can begin. Thirdly, inefficiently written software can significantly slow down simulation time or, worse, push users to seek solutions which are more complex and power-intensive (multithreading, GPUs, etc.) than the underlying task requires, decreasing reproducibility. Finally, even the relatively modest complexities of motion modelling in continuous environments raises the technical entry barrier to computational research and can impel researchers towards studying only one-dimensional environments or biologically unrealistic "gridworlds" with tabularised state spaces. Not only can gridworld models scale poorly in large environments but they typically disregard aspects of motion which can be non-trivial, for example speed variability and inertia.

Whilst there are valid reasons why gridworld and/or tabularised state-space models may be preferred – and good open source packages for modelling this (Chevalier-Boisvert et al. 2023; Juliani et al. 2022) – it is likely that coding simplicity, rather than theory-based justifications, remains a common reason these are used over continuous analogs. Recognizing these challenges, other recent efforts have sought to provide standardised frameworks for comparing different computational models against a library of experimental datasets (Dominé et al. 2024).

To overcome these issues, I built RatInABox (`https://github.com/RatInABox-Lab/RatInABox`): an open source Python toolkit for efficient and realistic motion modelling in complex continuous environments *and* concurrent simulation of neuronal activity data for many cell types including those typically found in the hippocampal formation (Figure 1.1).

1.2 RatInABox: Toolkit Components and Features

RatInABox is an open source software package comprising three component classes:

- `Environment`: The environment (or "box") that the `Agent` exists in. An `Environment` can be 1- or 2-dimensional, contain walls/barriers, holes, & objects and they can have periodic or solid boundary conditions (Figure 1.1a, b, d, and e).

- `Agent`: The agent (or "rat") moving around the `Environment` (Figure 1.1a and d). `Agent`s are 0-dimensional and `Environment`s can contain multiple `Agent`s simultaneously.

- `Neurons`: A population of neurons whose firing rates update to encode the "state" of the `Agent` in a rich variety of ways. Specific subclasses are provided corresponding to commonly studied cell-types (including, but not limited to, `PlaceCells`, `GridCells`, `BoundaryVectorCells` and

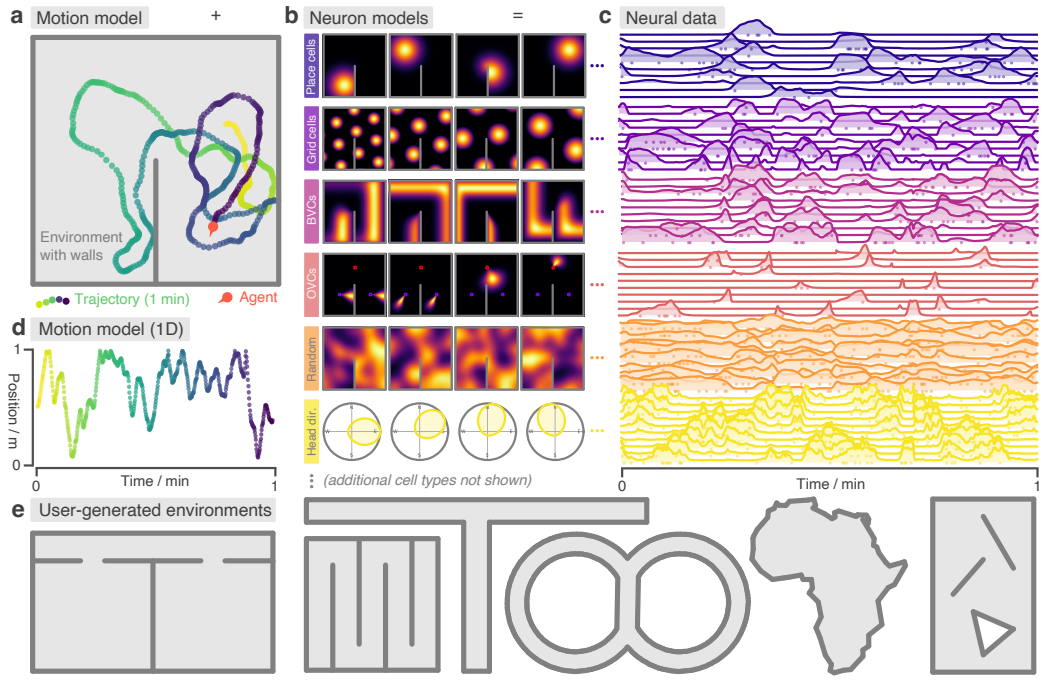

Figure 1.1: RatInABox is a flexible toolkit for simulating locomotion and neural data in complex continuous environments. **(a)** One minute of motion in a 2D `Environment` with a wall. By default the `Agent` follows a physically realistic random motion model fitted to experimental data. **(b)** Premade neuron models include the most commonly observed position/velocity selective cells types (6 of which are displayed here). Users can also build more complex cell classes based on these primitives. Receptive fields interact appropriately with walls and boundary conditions. **(c)** As the `Agent` explores the `Environment`, `Neurons` generate neural data. This can be extracted for downstream analysis or visualised using in-built plotting functions. Solid lines show firing rates, and dots show sampled spikes. **(d)** One minute of random motion in a 1D environment with solid boundary conditions. **(e)** Users can easily construct complex `Environments` by defining boundaries and placing walls, holes and objects. Six example `Environments`, some chosen to replicate classic experimental set-ups, are shown here.

`HeadDirectionCells`, Figure 1.1b and c). Users can also write their own `Neurons` subclasses or build/train complex function-approximator `Neurons` based on these primitives.

A typical workflow would be as follows: Firstly, an `Environment` is initialised with parameters specifying its dimensionality, size, shape and boundary conditions. Walls (or "barriers"), holes and objects (which act as "visual cues") can be added to make the `Environment` more complex. Secondly, an `Agent` is initialised with parameters specifying the characteristics of its

motion (mean/standard deviation of its speed and rotational velocity, as well as behaviour near boundaries). Thirdly, populations of `Neurons` are initialised with parameters specifying their characteristics (number of cells, receptive field parameters, maximum firing rates etc.).

Next, a period of simulated motion occurs: on each step the `Agent` updates its position and velocity within the `Environment`, given the duration of the step, and `Neurons` update their firing rates to reflect the new state of the `Agent`. After each step, data (timestamps, position, velocities, firing rates and spikes sampled according to an inhomogeneous Poisson process) are saved into their respective classes for later analysis; see fig. 1.1.

RatInABox is fundamentally continuous in space and time. Position and velocity are never discretised but are instead stored as continuous values and used to determine cell activity online, as exploration occurs. This differs from other models which are either discrete (e.g. "gridworld" or Markov decision processes) (Chevalier-Boisvert et al. 2023; Juliani et al. 2022) or approximate continuous rate maps using a cached list of rates precalculated on a discretised grid of locations (Cothi et al. 2020a). Modelling time and space continuously more accurately reflects real-world physics, making simulations smooth and amenable to fast or dynamic neural processes which are not well accommodated by discretised motion simulators. Despite this, RatInABox is still fast; to simulate 100 `PlaceCells` for 10 minutes of random 2D motion (dt = 0.1 s) it takes about 2 seconds on a consumer-grade CPU laptop (or 7 seconds for boundary vector cells).

By default the `Agent` follows a temporally continuous smooth random motion model, closely matched to the statistics of rodent foraging in an open field (Sargolini et al. 2006) (fig. 1.2); however, functionality is also provided for non-random velocity control (via a user-provided control signal) or for the `Agent` to follow an imported trajectory (fig. 1.3a). Once generated, data can be plotted using in-built plotting functions (which cover most of the figures in this chapter) or extracted to be used in the theoretical model being constructed by

the user.

1.2.1 Intended Use-Cases

Use cases are envisaged to fall into two broad categories. (i) Data generation: The user is interested in generating realistic trajectories and/or neural data for use in a downstream analysis or model training procedure (for example (Lee et al. 2023) or the work discussed in Chapter 5. (ii) Advanced modelling: The user is interested in building a model of the brain's role in navigation, including how behaviour and neural representations mutually interact (for example the work discussed in Chapter 4).

The most important details and features of RatInABox are briefly described below, divided into their respective classes. I leave all mathematical details to Appendix A. Additional details (including example scripts and figures) can also be found in Appendix A and on the GitHub repository. The codebase itself is comprehensively documented and can be referenced for additional understanding where necessary.

1.2.2 The `Environment`

Unlike discretised models, where environments are stored as sets of nodes ('states') connected by edges ('actions') (Juliani et al. 2022), here `Environment`s are continuous domains containing `walls` (1D line segments through which locomotion is not allowed) and `objects` (which are 0-dimensional and act as visual cues). Boundaries and visual cues are thought to provide an important source of sensory data into the hippocampus (O'Keefe et al. 1996; Hartley et al. 2000; Barry et al. 2006; Solstad et al. 2008) and play an important role in determining cell activity during navigation (Stachenfeld et al. 2017; Cothi et al. 2020a). An `Environment` can have periodic or solid boundary conditions and can be one- or two-dimensional (Figure 1.1a and d).

1.2.3 The `Agent`

Physically realistic random motion Smooth and temporally continuous random motion can be difficult to model. To be smooth (and therefore physically

plausible) a trajectory must be continuous in both position *and* velocity. To be temporally continuous, the statistics of the motion must be independent of the integration timestep being used. To be random, position and velocity at one time must not be reliable predictors of position and velocity at another time, provided these times are separated by a sufficiently long interval. Implementations of random motion models typically fail to satisfy one, or sometimes two, of these principles (Raudies et al. 2012; Benna et al. 2021).

Ornstein-Uhlenbeck processes, which sit at the heart of the RatInABox random motion model, are continuous-in-time random walks with a tendency to return to a central drift value. The decorrelation timescale can be also be controlled. These are used to update the velocity vector (linear and rotational velocities updated independently) on each update step. Position is then updated by taking a step along the velocity vector with some additional considerations to avoid walls. This method ensures both position and velocity are continuous, yet evolve "randomly" (fig. 1.1a and d), and the statistics of the motion is independent of the size of the discretisation timestep being used.

Reanalysing rat locomotion data from Sargolini et al. (2006) (as has been done before, by Raudies et al. (2012)) I found that the histograms of linear speeds are well fit by a Rayleigh distributions whereas rotational velocities are approximately fit by normal distributions (Figure 1.2a). Unlike Raudies et al. (2012), I also extract the decorrelation timescale of these variables and observe that rotational velocity in real locomotion data decorrelates nearly an order of magnitude faster than linear velocity (0.08 s vs. 0.7 s). The default parameters of the Ornstein-Uhlenbeck processes (including applying a transform on the linear velocity so its long-run distribution also follows a Rayleigh distribution, see Appendix A.2.1) are set to those measured from the Sargolini et al. (2006) dataset (Figure 1.2b).

Motion near walls Animals rarely charge head-first into a wall, turn around, then continue in the opposite direction. Instead, they slow down smoothly and turn to avoid a collision. Additionally, during random foraging, rodents

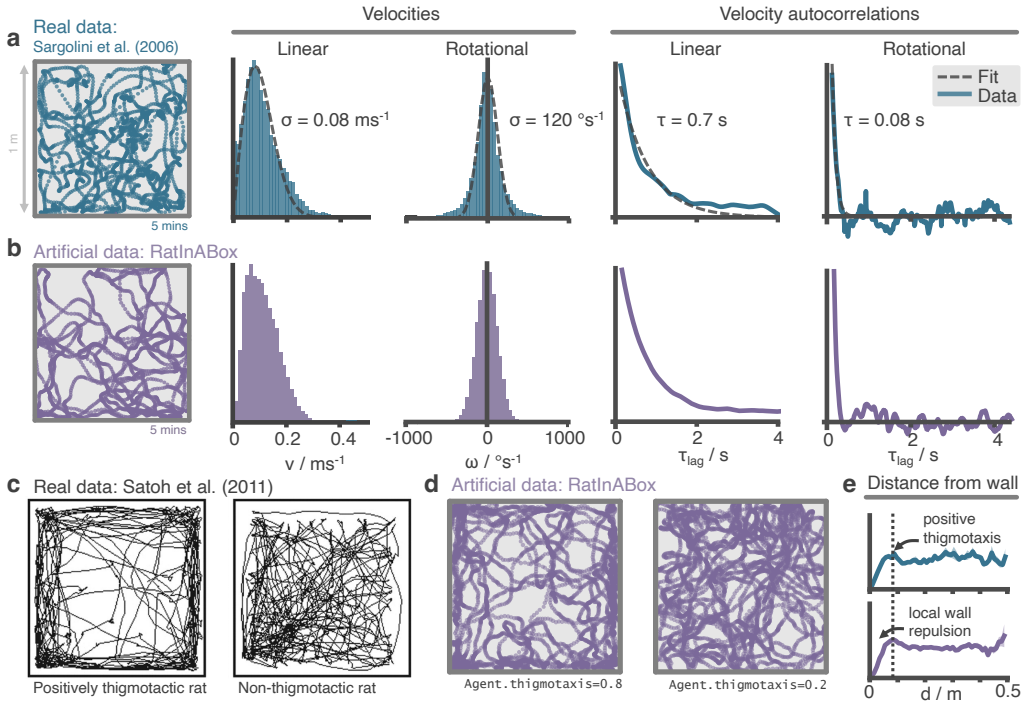

Figure 1.2: The RatInABox random motion model closely matches features of real rat locomotion. **(a)** An example 5 minute trajectory from the Sargolini et al. (2006) dataset. Linear velocity (Rayleigh fit) and rotational velocity (Gaussian fit) histograms and the temporal autocorrelations (exponential fit) of their time series'. **(b)** A sampled 5 minute trajectory from the RatInABox motion model with parameters matched to the Sargolini data. **(c)** Figure reproduced from Figure 8D (Satoh et al. 2011) showing 10 minutes of open-field exploration. "Thigmotaxis" is the tendency of rodents to over-explore near boundaries/walls and has been linked to anxiety. **(d)** RatInABox replicates the tendency of agents to over-explore walls and corners, flexibly controlled with a `'thigmotaxis'` parameter. **(e)** Histogram of the area-normalised time spent in annuli at increasing distances, d, from the wall. RatInABox and real data are closely matched in their tendency to over-explore locations near walls without getting too close.

are observed to show a bias towards following walls, a behaviour known as thigmotaxis (Satoh et al. 2011) (Figure 1.2c). To replicate these observations, walls in the `Environment` lightly repel the `Agent` when it is close. Coupled with the finite turning speed this creates (somewhat counter-intuitively) a thigmotactic effect where the agent over-explores walls and corners, matching what is observed in the data (fig. 1.2e). A user-defined parameter called `"thigmotaxis"` can be used to control the strength of this emergent effect (fig. 1.2d).

Imported trajectories RatInABox supports importing trajectory data which can be used instead of the inbuilt random motion model. Imported trajectory data points (which may be of low temporal-resolution) are interpolated using cubic splines and smoothly upsampled to user-define temporal precision (Figure 1.3a). This upsampling is essential if one wishes to use low temporal resolution trajectory data to generate high temporal resolution neural data.

Trajectory control RatInABox supports online velocity control. At each integration step a target drift velocity can be specified, towards which the `Agent` accelerates. I anticipate this feature being used to generate complex stereotyped trajectories or to model processes underpinning complex spatial behaviour (as I demonstrate in fig. 1.3b and e).

1.2.4 `Neurons`

RatInABox provides multiple premade `Neurons` subclasses chosen to replicate the most popular and influential cell models and state representations across computational neuroscience and machine learning. A selection of these are shown in Figure 1.1b. See Appendix A.2.3 for mathematical details. These currently include:

- `PlaceCells`: A set of locations is sampled uniformly at random from across the `Environment` or provided manually, each defining the centre of a place field. The place cell firing rate is determined by the some function of the distance from the `Agent` to the centre of the place field. Provided functions are
  - Gaussian: A Gaussian centred on the place field centre.
  - Gaussian threshold: A gaussian cropped and levelled at 1 standard deviation.
  - Difference of two Gaussians: A wide Gaussian subtracted from a narrower Gaussian with zero total volume.
  - Top hat: Fires uniformly only within a circle of specific radius (similar to tile coding in machine learning).

  - One hot: Only the closest place cell to a given position will fire. This is useful for replicating tabular state spaces but with continuous motion.
  - `PhasePrecessingPlaceCells`: A subclass of `PlaceCells` which display phase precession (O'Keefe et al. 1993) with respect to a background LFP theta-oscillation.

- `GridCells`: Grid cells are modelled using a method proposed by Burgess et al. (2007). Receptive fields are given by the thresholded or shifted sum of three cosine waves at 60°.

- `VectorCells`: Each vector cells responds to salient features in the `Environment` at a preferred distance and angle according to a model inspired by the double-Gaussian model used by Hartley et al. (2000). Vector cells can be "allocentric" (angular preferences are relative to true-North) or "egocentric" (Byrne et al. 2007) (angular preferences are relative to the `Agent`'s heading). Types include:
  - `BoundaryVectorCells`: Respond to `walls`
  - `ObjectVectorCells`: Respond to `objects`
  - `AgentVectorCells`: Respondto other `Agent`s
  - `FieldOfViewBVCs/OVCs/AVCs` (Egocentric vector cells arranged to tile the `Agent`'s field-of-view, further described below)

- `HeadDirectionCells`: Each cell has a preferred direction. The firing rate is given by a von Mises distribution centred on the preferred direction.

- `VelocityCells`: Like `HeadDirectionCells` but firing rate scales proportional to speed.

- `SpeedCell`: A single cell fires proportional to the scalar speed of the `Agent`.

- `RandomSpatialNeurons`: Each cell has a locally-smooth but random spatial receptive field of user-defined lengthscale.

A dedicated space containing additional cell classes not described here, is made available for community contributions to this list.

Customizable and trainable Neurons Any single toolkit cannot contain all possible neural representations of interest. Besides, static cell types (e.g. `PlaceCells`, `GridCells` etc.) which have fixed receptive fields are limiting if the goal is to study how representations and/or behaviour are learned. RatInABox provides two solutions: Firstly, being open-source, users can write and contribute their own bespoke `Neurons` (instructions and examples are provided) with arbitrarily complicated rate functions.

Secondly, two types of function-approximator `Neurons` are provided which map inputs (the firing rate of other `Neurons`) to outputs (firing rate) through a parameterised function which can be hand-tuned or trained to represent an endless variety of receptive field functions including those which are mixed selective, non-linear, dynamic and non-stationary.

- `FeedForwardLayer`: Calculates a weighted linear combination of the input `Neurons` with optional bias and non-linear activation function.

- `NeuralNetworkNeurons`: Inputs are passed through a user-provided artificial neural network.

Naturally, function-approximator `Neurons` can be used to model how neural populations in the brain communicate, how neural representations are learned or, in certain cases, neural dynamics. In an online demo I show how grid cells and head direction cells can be easily combined using a `FeedForwardLayer` to create head-direction selective grid cells (aka. conjunctive grid cells (Sargolini et al. 2006)). In Figure 1.3d and associated demo `GridCells` provide input to a `NeuralNetworkNeuron` which is then trained, on data generated during exploration, to have a highly complex and non-linear receptive field. Function-approximator `Neurons` can themselves be used as inputs to other function-

approximator `Neurons` allowing multi-layer and/or recurrent networks to be constructed and studied.

Field of view encodings Efficiently encoding what an `Agent` can 'see' in its local vicinity (aka. its field of view) is crucial for many modelling studies. A common approach is to use a convolutional neural network (CNN) to process an image of the nearby environment and extract activations from the final layer. However, this method is computationally expensive and necessitates training the CNN on a large dataset of visual images.

RatInABox offers a more efficient alternative through the use of `VectorCells`. Three variants – `FieldOfViewBVCs`, `FieldOfViewOVCs`, and `FieldOfViewAVCs` – comprise populations of *egocentric* `Boundary-`, `Object-`, and `AgentVectorCells` with angular and distance preferences specifically set to tile the `Agent`'s field of view. Being egocentric means that the cells remained fixed in the reference frame of the `Agent` as it navigates the `Environment`. Users define the range and resolution of this field of view. Plotting functions for visualising the field of view cells, as shown in Figure 1.3c, are provided.

Geometry and boundary conditions In RatInABox, `PlaceCells` and `VectorCells` are sensitive to walls in the `Environment`. Three distance geometries are supported: `'euclidean'` geometry calculates the Euclidean distance to a place field centre and so cell activity will 'bleed' through boundaries as if they weren't there. `'line_of_sight'` geometry allows a place cell to fire only if there is direct line-of-sight to the place field centre from the current location. Finally `'geodesic'` geometry (default) calculates distance according to the shortest boundary-avoiding path to the cell centre (notice smooth wrapping of the third place field around the wall in Figure 1.1b). The latter two geometries respect the observation that place fields don't typical pass through walls, an observation which is thought to support efficient generalisation in spatial reinforcement learning (Gustafson et al. 2011). Boundary conditions can be `periodic` or `solid`. In the former case, place fields corresponding to cells near the boundaries of the environment will wrap around.

Rate maps RatInABox simplifies the calculation and visualization of rate maps through built-in protocols and plotting functions. Rate maps can be derived explicitly from their known analytic firing functions or implicitly from simulation data. The explicit method computes rate maps by querying neuron firing rates at all positions simultaneously, utilizing 'array programming' to rapidly compute the rate map. In the implicit approach, rate maps are created by plotting a smoothed histogram of positions visited by the `Agent`, weighted by observed firing rates. Additionally, the tool offers the option to visualize spikes through raster plots.

1.3 Validation and Use-Case Demonstrations

The default parameters of the random motion model in RatInABox are matched to observed statistics of rodent locomotion, extracted by reanalysing data from Sargolini et al. (2006). Trajectories and statistics from the real data (Figure 1.2a) closely compare to the artificially generated trajectories from RatInABox (Figure 1.2b). Further, data (Satoh et al. 2011) shows that rodents have a tendency to over-explore walls and corners, a bias often called "thigmotaxis" which is particularly pronounced when the animal is new to the environment (Figure 1.2c). This bias is correctly replicated in the artificial trajectories generated by RatInABox - the strength of which can be controlled by a single parameter `Agent.thigmotaxis` (Figure 1.2d and e).

RatInABox can import and smoothly interpolate user-provided trajectory data. This is demonstrated in Figure 1.3a where a low-resolution trajectory is imported into RatInABox and smoothly upsampled using cubic spline interpolation. The resulting trajectory is a close match to the ground truth. Note that without upsampling, this data (2 Hz) would be far too low in temporal-resolution to usefully simulate neural activity. For convenience, the exact datafile (Sargolini et al. (2006)) used in Figure 1.3a and fig. 1.2a is uploaded with permission to the GitHub repository and can be imported using `Agent.import_trajectory(dataset="sargolini")`. An additional

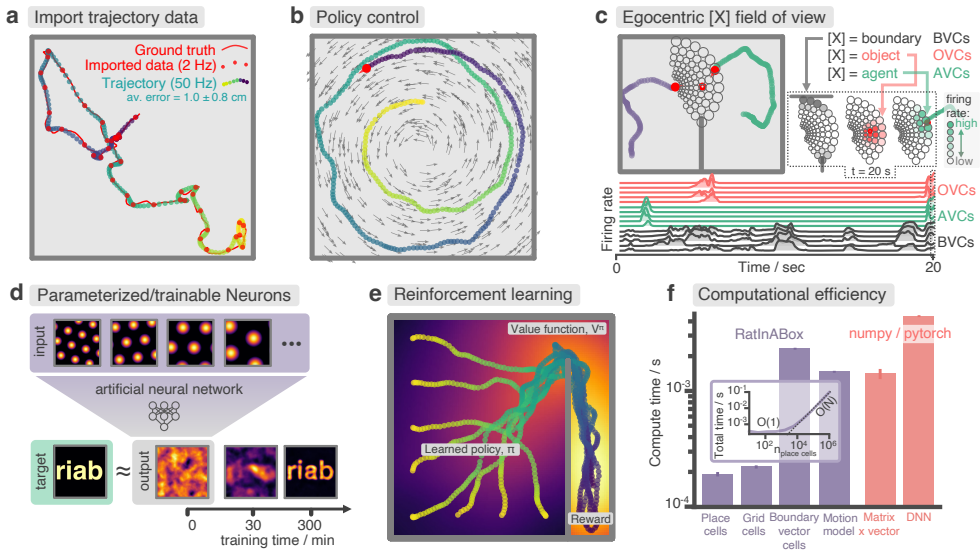

Figure 1.3: Advanced features and computational efficiency analysis. **(a)** Low temporal-resolution trajectory data (2 Hz) imported into RatInABox is upsampled ("augmented") using cubic spline interpolation. The resulting trajectory is a close match to the ground truth trajectory (Sargolini et al. (2006)) from which the low resolution data was sampled. **(b)** Movement can be controlled by a user-provided "drift velocity" enabling arbitrarily complex motion trajectories to be generated. Here I demonstrate how circular motion can be achieved by setting a drift velocity (grey arrows) which is tangential to the vector from the centre of the `Environment` to the `Agent`'s position. **(c)** Egocentric `VectorCells` can be arranged to tile the `Agent`'s field of view, providing an efficient encoding of what an `Agent` can 'see'. Here, two `Agents` explore an `Environment` containing walls and an object. `Agent`-1 (purple) is endowed with three populations of `Boundary`- (grey), `Object`- (red), and `Agent`- (green) selective field of view `VectorCells`. Each circle represents a cell, its position (in the head-centred reference frame of the `Agent`) corresponds to its angular and distance preferences and its shading denotes its current firing rate. The lower panel shows the firing rate of five example cells from each population over time. **(d)** A `Neurons` class containing a feedforward neural network learns, from data collect online over a period of 300 minutes, to approximate a complex target receptive field from a set of grid cell inputs. This demonstrates how learning processes can be incorporated and modelled into RatInABox. **(e)** RatInABox used in a simple reinforcement learning example. A policy iteration technique converges onto an optimal value function (heatmap) and policy (trajectories) for an `Environment` where a reward is hidden behind a wall. State encoding, policy control and the `Environment` are handled naturally by RatInABox. **(f)** Compute times for common RatInABox (purple) and non-RatInABox (red) operations on a consumer-grade CPU. Updating the random motion model and calculating boundary vector cell firing rates is slower than place or grid cells (note log-scale) but comparable, or faster than, size-matched non-RatInABox operations. Inset shows how the total update time (random motion model and place cell update) scales with the number of place cells.

trajectory dataset from a much larger environment is also supplied with permission from Tanni et al. (2022).

RatInABox is computationally efficient. I compare compute times for typical RatInABox operations (Figure 1.3f, purple bars) to typical *non*-RatInABox operations representing potential 'bottlenecking' operations in a downstream analysis or model-training procedure for which RatInABox is providing data (Figure 1.3f, red bars). These were multiplying a matrix by a vector using the `numpy` (Harris et al. 2020) package and a forward and backward pass through a small feedforward artificial neural network using the `pytorch` package (Paszke et al. 2019). `PlaceCells`, `GridCells` and the random motion model all update faster than these two operations. `BoundaryVectorCells` (because they require integrating around a 360° field-of-view) are significantly slower than the other cells but still outpace the feedforward neural network. All vector, matrix, and cell populations were size $n = 100$, the feedforward network had layer sizes $n_L = (100, 1000, 1000, 1)$, the `Environment` was 2D with no additional walls and all operations were calculated on a consumer-grade CPU (MacBook Pro, Apple M1). These results imply that, depending on the details of the use-case, RatInABox will likely not be a significant computational bottleneck.

Testing (Figure 1.3f, inset) reveals that the combined time for updating the motion model and a population of `PlaceCells` scales sublinearly $\mathsf{O}(1)$ for small populations $n < 1000$ where updating the random motion model dominates compute time, and linearly for large populations $n > 1000$. `PlaceCells`, `BoundaryVectorCells` and the `Agent` motion model update times will be additionally affected by the number of walls/barriers in the `Environment`. 1D simulations are significantly quicker than 2D simulations due to the reduced computational load of the 1D geometry.

1.3.1 Case Studies

I envisage RatInABox being used to support a range of theoretical studies by providing data and, if necessary, infrastructure for building models powered

by this data. This 'Bring-Your-Own-Algorithm' approach makes the toolkit generally applicable, not specialised to one specific field. Two exemplar use-cases are provided in Appendix A.3.2 and are briefly described below. The intention is to demonstrate the capacity of RatInABox for use in varied types of computational studies and to provide tutorials as a tool for learning how to use the package. Many more demonstrations and accompanying notebooks are provided on the Github repository.

In my first example I perform a simple experiment where location is decoded from neural firing rates (summarised in Figure A.1). Data – the location and firing rate trajectories of an `Agent` randomly exploring a 2D `Environment` – are generated using RatInABox. Gaussian process regression is used to predict position from firing rates on a held-out testing dataset. I compare the accuracy of decoding using different cell types; place cells, grid cells and boundary vector cells.

Next, I demonstrate the application of RatInABox to a simple reinforcement learning (RL) task (summarised in Figure 1.3e). A small network capable of model-free RL is constructed and trained using RatInABox. First a neuron calculates and learns – using a continuous variant of temporal difference learning – the value function $V^\pi(x) = \sum_i w_i F_i^{\text{pc}}(x)$ as a linear combination of place cell basis features. Then a new 'improved' policy is defined by setting a drift velocity – which biases the `Agent`'s motion – proportional to the gradient of the value function $v^{\text{drift}}(x) = \pi(x) \propto \nabla V^\pi|_x$. The `Agent` is therefore encouraged to move towards regions with high value. Iterating between these stages over many episodes ("policy iteration") results in convergence towards near optimal behaviour where the `Agent` takes the shortest route to the reward, avoiding the wall (Figure 1.3e).

Additional tutorials, not described here but available online, demonstrate how RatInABox can be used to model splitter cells, conjunctive grid cells, biologically plausible path integration, successor features, deep actor-critic RL, whisker cells and more. Despite including these examples I stress that

they are not exhaustive. RatInABox provides the framework and primitive classes/functions from which highly advanced simulations such as these can be built.

1.4 Discussion: Scope, Assumptions, and Future Development

RatInABox is a lightweight, open-source toolkit for generating realistic, standardised trajectory and neural data in continuous environments. It should be particularly useful to those studying spatial navigation and the role of the hippocampal formation. It remains purposefully small in scope - intended primarily as a means for generating data. I do not provide, nor intend to provide, a set of benchmark learning algorithms to use on the data it generates. Its user-friendly API, inbuilt data-plotting functions and general yet modular feature set mean it is well placed empower a wide variety of users to more rapidly build, train and validate models of hippocampal function (Lee et al. 2023) and spatial navigation (George et al. 2023b), accelerating progress in the field.

This package is not the first to model neural data (Stimberg et al. 2019; Hepburn et al. 2012; Hines et al. 1997) or spatial behaviour (Todorov et al. 2012; Merel et al. 2019) yet it distinguishes itself by integrating these two aspects within a unified, lightweight framework. The modelling approach employed by RatInABox involves certain assumptions:

1. It does not engage in the detailed exploration of biophysical (Stimberg et al. 2019; Hines et al. 1997) or biochemical (Hepburn et al. 2012) aspects of neural modelling, nor does it delve into the mechanical intricacies of joint and muscle modelling (Todorov et al. 2012; Merel et al. 2019). While these elements are crucial in specific scenarios, they demand substantial computational resources and become less pertinent in studies focused on higher-level questions about behaviour and neural representations.

2. A focus of this package is modelling experimental paradigms commonly used to study spatially modulated neural activity and behaviour in rodents. Consequently, environments are currently restricted to being two-dimensional and planar, precluding the exploration of three-dimensional settings. However, in principle, these limitations can be relaxed in the future.

3. RatInABox avoids the oversimplifications commonly found in discrete modelling, predominant in reinforcement learning (Chevalier-Boisvert et al. 2023; Juliani et al. 2022), which I believe impede its relevance to neuroscience.

4. Currently, inputs from different sensory modalities, such as vision or olfaction, are not explicitly considered. Instead, sensory input is represented implicitly through efficient allocentric or egocentric representations. If necessary, one could use the RatInABox API in conjunction with a third-party computer graphics engine to circumvent this limitation.

5. Finally, focus has been given to generating synthetic data from steady-state systems. Hence, by default, agents and neurons do not explicitly include learning, plasticity or adaptation. Nevertheless, it has been shown that a minimal set of features such as parameterised function-approximator neurons and policy control enable time varying behavioural policies and cell responses (Bostock et al. 1991; Barry et al. 2007) to be modelled within the framework.

In conclusion, while no single approach can be deemed the best, I believe that RatInABox's unique positioning makes it highly suitable for normative modelling and NeuroAI. I anticipate that it will complement existing toolkits and represent a significant contribution to the computational neuroscience toolbox.

Chapter 2

Theta Sequences as Eligibility Traces: A Biological Solution to Credit Assignment

Hippocampal RL, Part 1

This chapter was submitted to a short paper workshop, hence its brevity. It can be viewed as a theoretical distillation of the mechanisms and model more thoroughly explored in Chapter 3, and may be skipped entirely without loss of continuity.

Summary

Credit assignment problems, for example policy evaluation in RL, often require bootstrapping prediction errors through preceding states *or* maintaining temporally extended memory traces; solutions which are unfavourable or implausible for biological networks of neurons. I propose theta sequences – chains of neural activity during theta oscillations in the hippocampus, thought to represent rapid playthroughs of awake behaviour – as a solution. By analysing and simulating a model for theta sequences I show they compress behaviour such that existing but short $\mathsf{O}(10)$ ms neuronal memory traces are effectively extended allowing for bootstrap-free credit assignment without long memory

traces, equivalent to the use of eligibility traces in TD(λ).

2.1 Introduction: The Timescale Mismatch in Biological Reinforcement Learning

When one decodes position, x_E, from the hippocampus (HPC) of a rodent it sweeps from behind to in front of the *true* position, x_T, once every theta cycle (a strong 5-10 Hz neural oscillation). So-called "theta sequences" (Foster et al. 2007) don't make sense if the only goal of HPC is to accurately encode self-location at all times, they likely serve some other objective (Drieu et al. 2019). Building off a body of literature linking fast hippocampal phenomena to learning and RL (Mehta et al. 2000; Bono et al. 2021; George et al. 2023a), here it is demonstrated that theta sequences accelerate learning analogous to how eligibility traces (ETs) accelerate policy evaluation in RL. Policy evaluation with temporal difference (TD) learning permits two kinds of solutions: prediction errors can be bootstrapped through preceding states one-by-one (TD(0)) or temporally extended ETs can be maintained so credit can be assigned to states directly (Monte-Carlo, aka. TD(1)). These approaches are unified by the TD(λ) algorithm (Sutton 1988) (see Appendix B.2). Learning with long ETs, TD($\lambda > 0$), is typically faster, and therefore desirable, but biologically implausible since individual neurons have no trivial way to maintain ETs over timescales significantly longer than the membrane time constant $\mathsf{O}(10-50)$ ms. Perhaps theta sequences provide a solution to this problem: starting behind and moving in front of the animal rapidly within each cycle, the series of states observed within a sequence is an exact temporal compression of the states encountered on behavioural timescales (Figure 2.1a). In this regime the short neuronal ETs are magnified by the same compression factor and long ETs are indirectly achieved (see Appendix B.4). I derive the relationship between TD(λ) and theta sequences and empirically test it on a policy evaluation task (Figure 2.1b) by comparing artificial agents implementing TD(λ) with varying λ's (Figure 2.1c) to biological agents with short eligibility traces, TD($\lambda \approx 0$),

undergoing theta sequences of varying velocity (Figure 2.1d).

2.2 An Equivalence Between Theta Sequences and Eligibility Traces

Temporal difference learning using bioplausibly short ETs, $\tau_z = 10$ ms, on theta sequences is algorithmically equivalent to learning with long ETs τ_z^{eff} without theta sequences (see Appendix B). The effective compression is given by the ratio of the sequence velocity to the true agent velocity

$$\tau_z^{\text{eff}} = \frac{|\dot{x}_E|}{|\dot{x}_T|}\tau_z. \tag{2.1}$$

Agents move at a constant velocity of $v_T = 10$ cm s^{-1} around a 2 m track upon which a small reward is located, whilst learning the value function (Figure 2.1b). Increasing theta sequence velocity accelerates learning for the biological agent similar to how increasing the ET timescale accelerates learning for the artificial agent (Figure 2.1cd, top panel). When sequence velocity is low, learning resembles heavily bootstrapped TD(0) with the value function slowly creeping back from the reward site over time. When sequence velocity is high, learning resembles TD(1) with credit appropriately assigned to all states simultaneously (Figure 2.1cd, bottom panel). Biologically realistic sequence velocities (2 - 10 ms^{-1} (Wikenheiser et al. 2015)) match the range in the model where there is a sharp change from TD(0)-like to TD(1)-like learning regimes. Small errors can be observed in biological learning (Figure 2.1d doesn't converge to 1 for slower sequence speeds) due to, I suspect, 'loop effects' (Appendix Appendix B.5) occuring when the sequence discontinuously resets once per theta cycle. These loop effects are not catastrophic for learning. Despite these effects I actually find learning on theta sequences is overall *less* noisy (compare value estimates in Figure 2.1c and d) probably because, where the artificial agent can visit a location once per lap, theta sequences can traverse a location multiple times, smoothing learning. Learning with very fast sequences outpaces the artificial

equivalent, probably because a single sweep (the very first) can explore the entire environment whereas the sequence-less artificial agent must wait until at least one lap for it to have observed all states. In reality sweeps this fast are not observed in the brain.

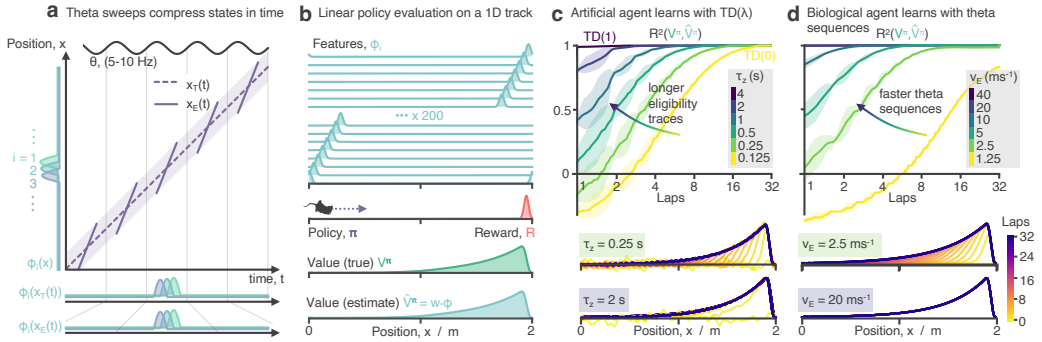

Figure 2.1: **a** Theta sequences: $x_E(t)$ (encoded position), sweeps from behind to in front of $x_T(t)$ (true position), compressing spatial inputs, therefore indirectly extending memory traces. **b** A policy evaluation task on a periodic 1D track. The value function is approximated as a linear sum of Gaussian basis features. **c** An artificial agent learns with TD(λ). (Top) Learning curves showing R^2 between true and estimated value functions for increasingly long eligibility traces (increasing λ). (Bottom) Evolution of the value estimate over learning for two opposing regimes: short eligibility traces (lots of bootstrapping) and long eligibility traces (no/little bootstrapping) **d** As in panel c but a biological agent with short eligibility traces 10 ms learns with theta sequences of increasing velocity. Sequence velocities are chosen to match eligibility timescales in panel c according to the proposed theory.

2.3 Discussion: Functional Implications

Theta sequences provide a viable mechanism by which biological networks of neurons can perform long-term credit assignment without resorting to slow bootstrapping nor maintaining implausibly long memory traces. Increasing sequence velocity is equivalent to increasing λ – using longer ETs – in TD(λ). Interestingly, in the brain theta power correlates with environmental uncertainty (Cavanagh et al. 2011) as well as periods of learning (Joensen et al. 2023) and sequence velocity depends on an animal's proximity to reward (Wikenheiser et al. 2015); based on the results shown here, I conjecture that top-down processes may actively control theta sequence speeds in order to accelerate or slow down learning depending on local conditions.

Chapter 3

Rapid Learning of Predictive Maps with STDP and Theta Phase Precession

Hippocampal RL, Part 2

Summary

Chapter 2 showed how theta phase precession can compress the timescale of behaviour allowing short timescale learning rules to rapidly learn long timescale associations. Here I build a biologically plausible model applying these ideas to learning successor features—representations encoding long timescale behavioural associations—with STDP—a short timescale learning rule for spiking neurons.

The predictive map hypothesis is a promising candidate principle for hippocampal function. A favoured formalisation of this hypothesis, called the successor representation, proposes that each place cell encodes the expected state occupancy of its target location in the near future. This predictive framework is supported by behavioural as well as electrophysiological evidence and has desirable consequences for both the generalisability and efficiency of

reinforcement learning algorithms. However, it is unclear how the successor representation might be learnt in the brain. Error-driven temporal difference learning, commonly used to learn successor representations in artificial agents, is not known to be implemented in hippocampal networks. Instead, it is demonstrated that spike-timing dependent plasticity (STDP), a form of Hebbian learning, acting on temporally compressed trajectories known as "theta sweeps", is sufficient to rapidly learn a close approximation to the successor representation. The model is biologically plausible – it uses spiking neurons modulated by theta-band oscillations, diffuse and overlapping place cell-like state representations, and experimentally matched parameters. I show how this model maps onto known aspects of hippocampal circuitry and explains substantial variance in the temporal difference successor matrix, consequently giving rise to place cells that demonstrate experimentally observed successor representation-related phenomena including backwards expansion on a 1D track and elongation near walls in 2D. Finally, the model provides insight into the observed topographical ordering of place field sizes along the dorsal-ventral axis by showing this is necessary to prevent the detrimental mixing of larger place fields, which encode longer timescale successor representations, with more fine-grained predictions of spatial location.

3.1 Introduction: The Hippocampus, Predictive Maps, and a Role for STDP

Knowing where you are and how to navigate in your environment is an everyday existential challenge for motile animals. In mammals, a key brain region supporting these functions is the hippocampus (Scoville et al. 1957; Morris et al. 1982), which represents self-location through the population activity of place cells – pyramidal neurons with spatially selective firing fields (O'Keefe et al. 1971). Place cells, in conjunction with other spatially tuned neurons (Taube et al. 1990; Hafting et al. 2005), are widely held to constitute a "cognitive map" encoding information about the relative location of remembered locations and

providing a basis upon which to flexibly navigate (Tolman 1948; O'Keefe et al. 1978).

The hippocampal representation of space incorporates spike time and spike rate based encodings, with both components conveying broadly similar levels of information about self-location (Skaggs et al. 1996b; Huxter et al. 2003). Thus, the position of an animal in space can be accurately decoded from place cell firing rates (Wilson et al. 1993) as well as from the precise time of these spikes relative to the background 8-10Hz theta oscillation in the hippocampal local field potential (Huxter et al. 2003). The latter is made possible since place cells have a tendency to spike progressively earlier in the theta cycle as the animal traverses the place field - a phenomenon known as phase precession (O'Keefe et al. 1993). Therefore, during a single cycle of theta the activity of the place cell population smoothly sweeps from representing the past to representing the future position of the animal (Maurer et al. 2006), and can simulate alternative possible futures across multiple cycles (Johnson et al. 2007).

In order for a cognitive map to support planning and flexible goal-directed navigation it should incorporate information about the overall structure of space and the available routes between locations (Tolman 1948; O'Keefe et al. 1978). Theoretical work has identified the regular firing patterns of entorhinal grid cells with the former role, providing a spatial metric sufficient to support the calculation of navigational vectors (Bush et al. 2015; Banino et al. 2018). In contrast, associative place cell-place cell interactions have been repeatedly highlighted as a plausible mechanism for learning the available transitions in an environment (Muller et al. 1991; Blum et al. 1996; Mehta et al. 2000). In the hippocampus, such associative learning has been shown to follow a spike-timing dependent plasticity (STDP) rule (Bi et al. 1998) – a form of Hebbian learning where the temporal ordering of spikes between presynaptic and postsynaptic neurons determines whether long-term potentiation or depression occurs. One of the consequences of phase precession is that correlates of behaviour, such as position in space, are compressed onto the timescale of a single theta cycle

and thus coincide with the time-window of STDP $\mathcal{O}(20-50$ ms) (Skaggs et al. 1996b; Mehta et al. 2000; Mehta 2001; Mehta et al. 2002). This combination of theta sweeps and STDP has been applied to model a wide range of sequence learning tasks (Jensen et al. 1996; Koene et al. 2003; Reifenstein et al. 2021), and as such, potentially provides an efficient mechanism to learn from an animal's experience – forming associations between cells which are separated by behavioural timescales much larger than that of STDP.

Spatial navigation can readily be understood as a reinforcement learning problem - a framework which seeks to define how an agent should act to maximise future expected reward (Sutton et al. 1998). Conventionally the value of a state is defined as the expected cumulative reward that can be obtained from that location with some temporal discount applied. Thus, the relationship between states and the rewards expected from those states are captured in a single value which can be used to direct reward-seeking behaviour. However, the computation of expected reward can be decomposed into two components – the successor representation, a predictive map capturing the expected location of the agent discounted into the future, and the expected reward associated with each state (Dayan 1993). Such segregation yields several advantages since information about available transitions can be learnt independently of rewards and thus changes in the locations of rewards do not require the value of all states to be re-learnt. This recapitulates a number of long-standing theories of hippocampus which state that hippocampus provides spatial representations that are independent of the animal's particular goal and support goal-directed spatial navigation (Redish et al. 1998; Burgess et al. 1997; Koene et al. 2003; Hasselmo et al. 2005; Erdem et al. 2012).

A growing body of empirical and theoretical evidence suggests that the hippocampal spatial code functions as a successor representation (Stachenfeld et al. 2017). Specifically, that the activity of hippocampal place cells encodes a predictive map over the locations the animal expects to occupy in the future. Notably, this framework accounts for phenomena such as the skewing of place

fields due to stereotyped trajectories (Mehta et al. 2000), the reorganisation of place fields following a forced detour (Alvernhe et al. 2011), and the behaviour of humans and rodents whilst navigating physical, virtual and conceptual spaces (Momennejad et al. 2017; Cothi et al. 2022b). However, the successor representation is typically conceptualised as being learnt using the temporal difference learning rule (Russek et al. 2017; Cothi et al. 2020b), which uses the prediction error between expected and observed experience to improve the predictions. Whilst correlates of temporal difference learning have been observed in the striatum during reward-based learning (Schultz et al. 1997), it is less clear how it could be implemented in the hippocampus to learn a predictive map. In this context, it was hypothesised that the predictive and compression properties of theta sweeps, combined with STDP in the hippocampus, might be sufficient to approximately learn a successor representation.

I simulated the synaptic weights learnt due to STDP between a set of synthetic spiking place cells and show they closely resemble the weights of a successor representation learnt with temporal difference learning. I found that the inclusion of theta sweeps with the STDP rule increased the efficiency and robustness of the learning, with the STDP weights being a close approximation to the temporal difference successor matrix. Further, I find no fine tuning of parameters is needed - biologically determined parameters are optimal to efficiently approximate a successor representation and replicate experimental results synonymous with the predictive map hypothesis, including the behaviourally biased skewing of place fields (Mehta et al. 2000; Stachenfeld et al. 2017) in realistic 1- and 2-dimensional environments. Finally, I use the simulation of STDP with theta sweeps to generate insight into the observed topographical ordering of place field sizes along the dorsal-ventral hippocampal axis (Kjelstrup et al. 2008), by observing that such organisation is necessary to prevent the detrimental mixing of larger place fields, which approximate longer timescale successor representations (Momennejad et al. 2018), with more fine-grained predictions of spatial location. This model, focussing on the

role of theta sweeps and STDP in learning a hippocampal predictive map, is part of a growing body of recent work emphasising hippocampally-plausible mechanisms of learning successor representations, such as using hippocampal recurrence (Fang et al. 2022) or synaptic learning rules which bootstrap long-range predictive associations (Bono et al. 2021).

3.2 STDP and Phase Precession Approximate the Successor Representation

The goal was to investigate whether a combination of STDP and phase precession is sufficient to generate a successor representation-like matrix of synaptic weights between place cells in CA3 and downstream CA1. The model comprises of an agent exploring a maze where its position $\mathbf{x}(t)$ is encoded by the instantaneous firing of a population of N CA3 basis features, each with a spatial receptive field $f_j^x(\mathbf{x})$ given by a thresholded Gaussian of radius 1 m and 5 Hz peak firing rate. As the agent traverses the receptive field, its rate of spiking is subject to phase precession $f_j^\theta(\mathbf{x}, t)$ with respect to a 10 Hz theta oscillation. This is implemented by modulating the firing rate by an independent phase precession factor which varies according to the current theta phase and how far through the receptive field the agent has travelled (Chadwick et al. 2015)(see Appendix C.3 and Figure 3.1a) such that, in total, the instantaneous firing rate of the j^{th} basis features is given by:

$$f_j(\mathbf{x}, t) = f_j^x(\mathbf{x}) f_j^\theta(\mathbf{x}, t). \tag{3.1}$$

CA3 basis features f_j then linearly drive downstream CA1 'STDP successor features' $\tilde{\psi}_i$ (Figure 3.1b)

$$\tilde{\psi}_i(\mathbf{x}, t) = \sum_j \mathsf{W}_{ij} f_j(\mathbf{x}, t). \tag{3.2}$$

Using an inhomogeneous Poisson process, the firing rates of the basis and STDP successor features are converted into spike trains which cause learning in the weight matrix W_{ij} according to an STDP rule (see Appendix C.4 and Figure 3.1c). The STDP synaptic weight matrix W_{ij} (Figure 3.1d) can then be directly compared to the temporal difference (TD) successor matrix M_{ij} (Figure 3.1e), learnt via TD learning on the CA3 basis features (the full learning rule is derived in Appendix C.5 and shown in Equation (C.24)). Further, the TD successor matrix M_{ij} can also be used to generate the 'TD successor features':

$$\psi_i(\mathbf{x}) = \sum_j \mathsf{M}_{ij} f_j^x(\mathbf{x}), \tag{3.3}$$

allowing for direct comparison and analyses with the STDP successor features $\tilde{\psi}_i$ (Equation (3.2)), using the same underlying firing rates driving the TD learning to sample spikes for the STDP learning. This abstraction of biological detail avoids the challenges and complexities of implementing a fully spiking network, although an avenue for correcting this would be the approach of Brea et al. (2016) and Bono et al. (2021). In my model phase precession generates theta sweeps (Figure 3.1a, grey box) as cells successively visited along the current trajectory fire at progressively later times in each theta cycle. Theta sweeps take the current trajectory of the agent and effectively compress it in time. As is shown below, these compressed trajectories are important for learning successor features.

3.2.1 STDP Approximates TD-learning

I first simulated an agent with $N = 50$ evenly spaced CA3 place cell basis features on a 5 m circular track (linear track with circular boundary conditions to form a closed loop, Figure 3.2a). The agent moved left-to-right at a constant velocity for 30 minutes, performing ∼58 complete traversals of the loop. The STDP weights learnt between the phase precessing basis features and their downstream STDP successor features (Figure 3.2b) were markedly similar to the successor representation matrix generated using temporal difference learning

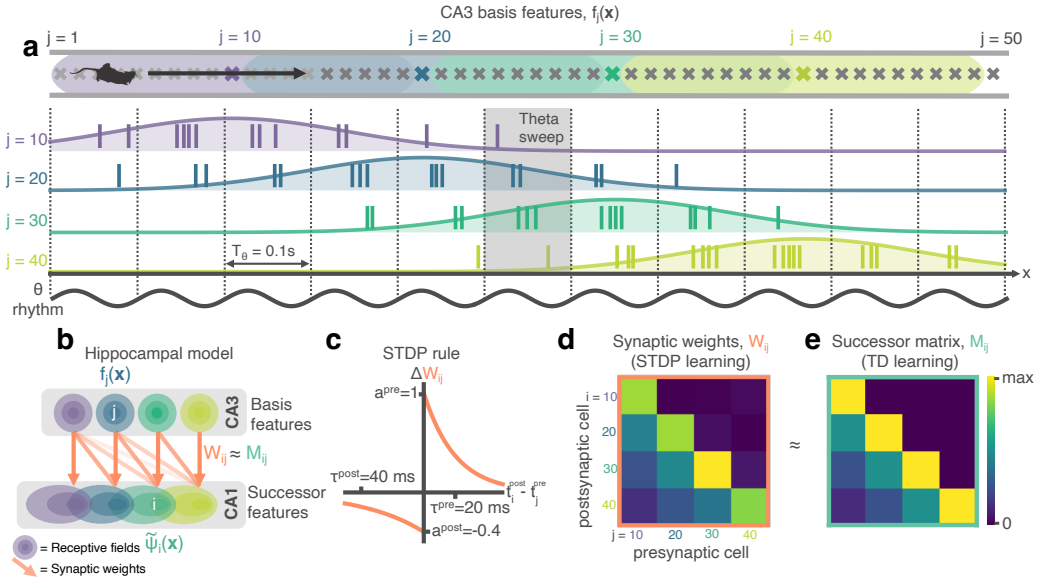

Figure 3.1: STDP between phase precessing place cells produces successor representation-like weight matrices. **a** Schematic of an animal running left-to-right along a track. 50 cells phase precess, generating theta sweeps (e.g. grey box) that compress spatial behaviour into theta timescales (10 Hz). **b** I simulate a population of CA3 'basis feature' place cells which linearly drive a population of CA1 'STDP successor feature' place cells through the synaptic weight matrix W_{ij}. **c** STDP learning rule; pre-before-post spike pairs ($t_i^{\text{post}} - t_j^{\text{pre}} > 0$) result in synaptic potentiation whereas post-before-pre pairs ($t_i^{\text{post}} - t_j^{\text{pre}} < 0$) result in depression. Depression is weaker than potentiation but with a longer time window, as observed experimentally. **d** Simplified schematic of the resulting synaptic weight matrix, W_{ij}. Each postsynaptic cell (row) fires just after, and therefore binds strongly to, presynaptic cells (columns) located to the left of it on the track. **e** Simplified schematic of the successor matrix (Equation (3.3)) showing the synaptic weights after training with a temporal difference learning rule, where each CA1 cell converges to represent the successor feature of its upstream basis feature. Backwards skewing (successor features "predict" upcoming activity of their basis feature) is reflected in the asymmetry of the matrix, where more activity is in the lower triangle, similar to panel d.

applied to the same basis features under the same conditions (Figure 3.2c, element-wise Pearson correlation between matrices $R^2 = 0.87$). In particular, the agent's strong left-to-right behavioural bias led to the characteristic asymmetry in the STDP weights predicted by successor representation models (Stachenfeld et al. 2017), with both matrices dominated by a wide band of positive weight shifted left of the diagonal and negative weights shifted right.

To compare the structure of the STDP weight matrix W_{ij} and TD successor matrix M_{ij}, I aligned each row on the diagonal and averaged across rows (see Appendix C.8), effectively calculating the mean distribution of learnt weights originating from each basis feature (Figure 3.2d). Both models exhibited a similar distribution, with values smoothly ramping up to a peak just left of centre, before a sharp drop-off to the right caused by the left-to-right bias in the agent's behaviour. In the network trained by TD learning this is because CA3 place cells to the left of (i.e. preceding) a given basis feature are reliable predictors of that basis feature's future activity, with those immediately preceding it being the strongest predictors and thus conferring the strongest weights to its successor feature. Conversely, the CA3 place cells immediately to the right of (i.e. after) this basis feature are the furthest they could possibly be from predicting its future activity, resulting in minimal weight contributions. Indeed, I observed some of these weights even becoming negative (Figure 3.2d) – necessary to approximate the sharp drop-off in predictability using the smooth Gaussian basis features. With the STDP model, the similar distribution of weights is caused by the asymmetry in the STDP learning rule combined with the consistent temporal ordering of spikes in a theta sweep. Hence, the sequence of spikes emitted by different cells within a theta cycle directly reflects the order in which their spatial fields are encountered, resulting in commensurate changes to the weight matrix. So, for example, if a postsynaptic neuron reliably precedes its presynaptic cell on the track, the corresponding weight will be reduced, potentially becoming negative. I note that weights changing their sign is not biologically plausible, as it is a violation of Dale's Law (Dale 1935).

This could perhaps be corrected with the addition of global excitation or by recruiting inhibitory interneurons.

Notably, the temporal compression afforded by theta phase precession, which brings behavioural effects into the millisecond domain of STDP, is an essential element of this process (Lisman et al. 2005; Koene et al. 2003). When phase precession was removed from the STDP model, the resulting weights failed to capture the expected behavioural bias and thus did not resemble the successor matrix - evidenced by the lack of asymmetry (Figure 3.2d, dashed line; ratio of mass either side of y-axis 4.54 with phase precession vs. 0.99 without) and a decrease in the explained variance of the TD successor matrix (Figure 3.2e, $R^2 = 0.87 \pm 0.01$ vs $R^2 = 0.63 \pm 0.02$ without phase precession). Similarly, without the precise ordering of spikes, the learnt weight matrix was less regular, having increased levels of noise, and converged over 4.5× more slowly (Figure 3.2e; time to reach $R^2 = 0.5$: 2.5 vs. 11.5 minutes without phase precession), still yet to fully converge over the course of 1 hour (Figure C.1–supplement 1a). Thus, the ability to approximate TD learning appears specific to the combination of STDP and phase precession. Indeed, there are deep theoretical connections linking the two - see Appendix C.9.1 for a theoretical investigation into the connections between TD learning and STDP learning augmented with phase precession. This effect is robust to variations in running speed (Figure C.1–supplement 1b) and field sizes (Figure C.1–supplement 1c), as well as scenarios where target CA1 cells have multiple firing fields (Figure C.2–supplement 2a) that are updated online during learning (Figure C.2–supplement 2b-d), or fully-driven by spikes in CA3 (Figure C.2–supplement 2e).

A hyperparameter sweep was also conducted to test if these results were robust to changes in the phase precession and STDP learning rule parameters (Figure C.3). The sweep range for each parameter contained and extended beyond the "biologically plausible" values used in this chapter (Figure C.3a). It was found that optimised parameters (those which result in the highest final similarity between STDP and TD weight matrices, W_{ij} and M_{ij}) were very close

to the biological parameters already selected for my model from a literature search (Figure C.3cd, parameter references also listed in figure) and, when they were used, no drastic improvement was seen in the similarity between W_{ij} and M_{ij}. The only exception was firing rate for which performance monotonically improved as it increased - something the brain likely cannot achieve due to energy constraints. In particular, the parameters controlling phase precession in the CA3 basis features (Figure C.4a) can affect the CA1 STDP successor features learnt, with 'weak' phase precession resembling learning in the absence of theta modulation (Figure C.4bc), biologically plausible values providing the best match to the TD successor features (Figure C.4d) and 'exaggerated' phase precession actually hindering learning (Figure C.4e). Additionally, I find these CA1 cells go on to inherit phase precession from the CA3 population even after learning when they are driven by multiple CA3 fields (Figure C.4f), and that this learning is robust to realistic phase offsets between the populations of CA3 and CA1 place cells (Figure C.4g).

Next, I examined the correspondence between my model and the TD-trained successor representation in a situation without a strong behavioural bias. Thus, I reran the simulation on the linear track without the circular boundary conditions so the agent turned and continued in the opposite direction whenever it reached each end of the track (Figure 3.2f). Again, the STDP and TD successor representation weight matrices where remarkably similar ($R^2 = 0.88$; Figure 3.2gh) both being characterised by a wide band of positive weight centred on the diagonal (Figure 3.2i) - reflecting the directionally unbiased behaviour of the agent. In this unbiased regime, theta sweeps were less important though still confered a modest shape, learning speed, and signal-strength advantage over the non-phase precessing model (Figure 3.2j) - evidenced as an increased amount of explained variance ($R^2 = 0.88 \pm 0.01$ vs. $R^2 = 0.76 \pm 0.02$) and faster convergence (time to reach $R^2 = 0.5$; 3 vs 7.5 minutes).

To test if the STDP model's ability to capture the successor matrix would

scale up to open field spaces, I implemented a 2D model of phase precession (see Appendix C.3) where the phase of spiking is sampled according to the distance travelled through the place field along the chord currently being traversed (Jeewajee et al. 2014). I then simulated the agent in an environment consisting of two interconnected 2.5 × 2.5 m square rooms (Figure 3.2k) using an adapted policy modelling rodent foraging behaviour that is biased towards traversing doorways and following walls (Raudies et al. 2012) (see Appendix C.7 and 10 minute sample trajectory shown in Figure 3.2k). After training for 2 hours of exploration, I found that the combination of STDP and phase precession was able to successfully capture the structure in the TD successor matrix (Figure 3.2l-m, $R^2 = 0.74$, TD successor matrix calculated over the same 2 hour trajectory).

3.2.2 Learned Place Fields Exhibit Behaviorally-Biased Skewing

The next step was to investigate how the similarities in weights between the STDP and TD successor representation models are conveyed in the downstream CA1 successor features. One hallmark of the successor representation is that strong biases in behaviour (for example, travelling one way round a circular track) induce a reliable predictability of upcoming future locations, which in turn causes a backward skewing in the resulting successor features (Stachenfeld et al. 2017). Such skewing, opposite to the direction of travel, has also been observed in hippocampal place cells (Mehta et al. 2000). Under strongly biased behaviour on the circular linear track, the biologically plausible STDP CA1 successor features (Equation (3.2)) had a very high correlation with the TD successor features (Equation (3.3)) predicted by successor theory (Figure 3.3a; $R^2 = 0.98 \pm 0.01$). Both exhibited a pronounced backward skew, opposite to the direction of travel (mean TD vs. STDP successor feature skewness: $= -0.39 \pm 0.01$ vs. $= -0.24 \pm 0.07$). Furthermore, both the STDP and TD successor representation models predict that such biased behaviour should induce a backwards shift in the location of place field peaks (Figure 3.3a

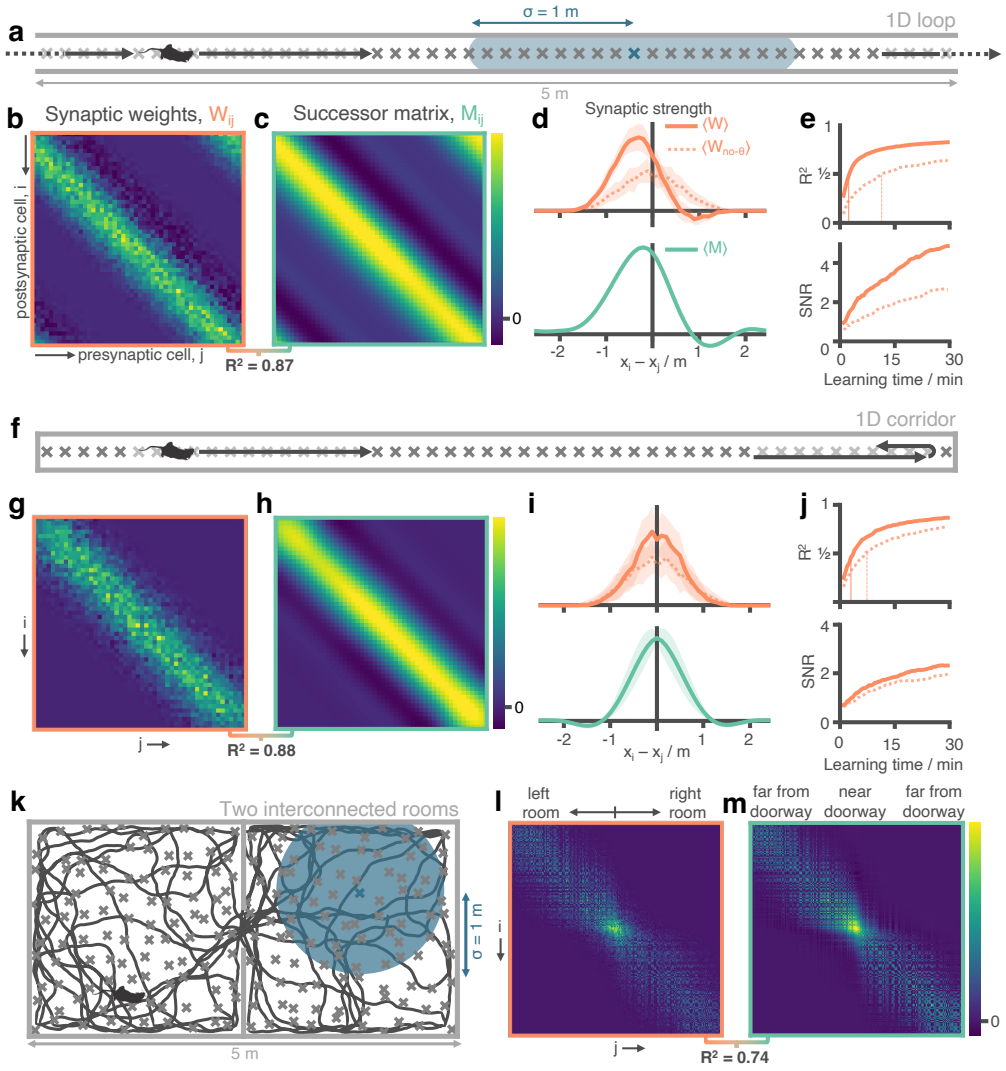

Figure 3.2: Successor matrices are rapidly approximated by STDP applied to spike trains of phase precessing place cells. **a** Agents traversed a 5 m circular track in one direction (left-to-right) with 50 evenly distributed CA3 spatial basis features (example thresholded Gaussian place field shown in blue, radius $\sigma = 1$ m). **b&c** After 30 minutes, the synaptic weight matrix learnt between CA3 basis features and CA1 successor features strongly resembles the equivalent successor matrix computed by temporal difference learning. Rows correspond to CA1, columns to CA3. **d** To compare the distribution of weights, matrix rows were aligned on the diagonal and averaged over rows (mean ± standard deviation shown). **e** Against training time, I plot (top) the R^2 between the synaptic weight matrix and successor matrix and (bottom) the signal-to-noise ratio of the synaptic matrix. Vertical lines show time where R^2 reaches 0.5. **f-j** Same as panels a-e except the agent turns around at each end of the track. The average policy is now unbiased with respect to left and right, as can be seen in the diagonal symmetry of the matrices. **k-m** As in panels a-c except the agent explores a two dimensional maze where two rooms are joined by a doorway. The agent follows a random trajectory with momentum and is biased to traverse doorways and follow walls.

left panel; TD vs. STDP successor feature shift in metres: -0.28 ± 0.00 vs -0.38 ± 0.03) – this phenomenon is also observed in the hippocampal place cells (Mehta et al. 2000), and my model accounts for the observation that more shifting and skewing is observed in CA1 place cells than CA3 place cells (Dong et al. 2021). As expected, when theta phase precession was removed from the model no significant skew or shift was observed in the STDP successor features. Similarly, the skew in field shape and shift in field peak were not present when the behavioural bias was removed (Figure 3.3b) – in this unbiased scenario, the advantage of the STDP model with theta phase precession was modest relative to the same model without phase precession ($R^2 = 0.99 \pm 0.01$ vs. $R^2 = 0.96 \pm 0.01$).

Examining the activity of CA1 cells in the two-room open field environment, I found an increase in the eccentricity of fields close to the walls (Figure 3.3c & d; average eccentricity of STDP successor features near vs. far from wall: 0.57 ± 0.06 vs. 0.33 ± 0.07). In particular, this increased eccentricity is facilitated by a shorter field width along the axis perpendicular to the wall (Figure 3.3e), an effect observed experimentally in rodent place cells (Tanni et al. 2021). This increased eccentricity of cells near the wall remained when the behavioural bias to follow walls was removed (Figure 3.3d; average eccentricity with vs. without wall bias: 0.57 ± 0.06 vs. 0.54 ± 0.06), thus indicating it is primarily caused by the inherent bias imposed on behaviour by extended walls rather than an explicit policy bias. Note that the ellipse fitting algorithm accounts for portions of the field that have been cut off by environmental boundaries (see Appendix C & Figure 3.3c), and so this effect is not simply a product of basis features being occluded by walls.

In a similar fashion, the bias in the motion model used here - which is predisposed to move between the two rooms - resulted in a shift in STDP successor feature peaks towards the doorway (Figure 3.3f & g; inwards shift in metres for STDP successor features near vs. far from doorway: 0.15 ± 0.06 vs. 0.04 ± 0.05; with doorway bias turned off: 0.05 ± 0.08 vs. 0.04 ± 0.05). At the

level of individual cells this was visible as an increased propensity for fields to extend into the neighbouring room after learning (Figure 3.3h). Hence, although basis features were initialised as two approximately non-overlapping populations – with only a small proportion of cells near the doorway extending into the neighbouring room – after learning many cells bind to those on the other side of the doorway, causing their place fields to diffuse through the doorway and into to the other room (Figure 3.3f). This shift could partially explain why place cell activity is found to cluster around doorways (Spiers et al. 2015) and rewarded locations (Dupret et al. 2010) in electrophysiological experiments. Equally it is plausible that a similar effect might underlie experimental observations that neural representations in multi-compartment environments typically begin heavily fragmented by boundaries and walls but, over time, adapt to form a smooth global representations (e.g., as observed in grid cells by Carpenter et al. (2015)).

3.2.3 Anatomical Segregation Supports Multiscale Predictive Maps

The final investigation concerned whether the STDP learning rule was able to form successor representation-like connections between basis features of different scales. Recent experimental work has highlighted that place fields form a multiscale representation of space, which is particularly noticeable in larger environments (Tanni et al. 2021; Eliav et al. 2021), such as the one modelled here. Such multiscale spatial representations have been hypothesised to act as a substrate for learning successor features with different time horizons – large scale place fields are able to make predictions of future location across longer time horizons, whereas place cells with smaller fields are better placed to make temporally fine-grained predictions. Agents could use such a set of multiscale successor features to plan actions at different levels of temporal abstraction, or predict precisely which states they are likely to encounter soon (Momennejad et al. 2018). Despite this, what is not known is whether different sized place fields will form associations when subject to STDP coordinated by

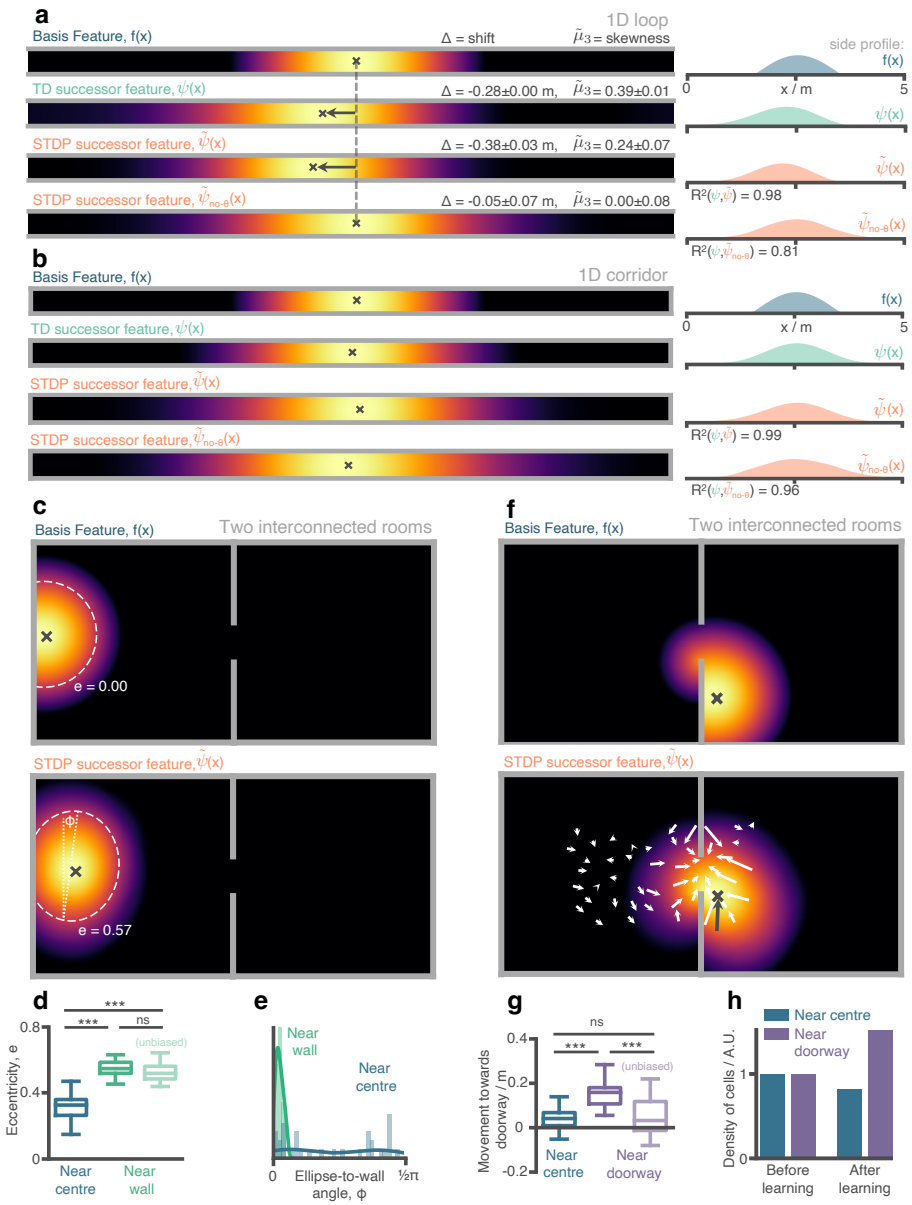

Figure 3.3: Place cells (aka. successor features) in my STDP model show behaviourally biased skewing resembling experimental observations and successor representation predictions. **a** In the loop maze (motion left-to-right) STDP place cells skew and shift backwards, and strongly resemble place cells obtained via temporal difference learning. This is not the case when theta phase precession is absent. **b** In the corridor maze, where travel in either direction is equally likely, place fields diffuse in both directions due to the unbiased movement policy. **c** In the 2D maze, place cells (of geodesic Gaussian basis features) near the wall elongate along the wall axis (dashed line shows best fitting ellipse, angle construct show the ellipse-to-wall angle). **d** Place cells near walls have higher elliptical eccentricity than those near the centre of the environments. This increase remains even when the movement policy bias to follow walls is absent. **e** The eccentricity for fields near the walls is facilitated by an increase in the length of the place field along an axis parallel to the wall (ϕ close to zero). *(Continued on next page...)*

Figure 3.3: *(...continued from previous page)* **f** Place cells near the doorway cluster towards it and expand through the doorway relative to their parent basis features. **g** The shift of place fields near the doorway towards the doorway is significant relative to place fields near the centre and disappears when the behavioural bias to cross doorways is absent. **h** The shift of place fields towards the doorway manifests as an increase in density of cells near the doorway after exploration.

phase precession and what effect this would have on the resulting successor features. Hypothetically, consider a small basis feature cell with a receptive field entirely encompassed by that of a larger basis cell with no theta phase offset between the entry points of both fields. A potential consequence of theta phase precession is that the cell with the smaller field would phase precess faster through the theta cycle than the other cell - initially it would fire later in the theta cycle than the cell with a larger field, but as the animal moves towards the end of the small basis field it would fire earlier. These periods of potentiation and depression instigated by STDP could act against each other, and the extent to which they cancel each other out would depend on the relative placement of the two fields, their size difference, and the parameters of the learning rule. To test this, I simulated an agent, learning according to my STDP model in the circular track environment, with, simultaneously, three sets of differently sized basis features ($\sigma = 0.5$, 1.0 and 1.5 m, Figure 3.4a). Such ordered variation in field size has been observed along the dorso-ventral axis of the hippocampus (Kjelstrup et al. 2008; Strange et al. 2014); Figure 3.4b), and has been theorised to facilitate successor representation predictions across multiple time-scales (Stachenfeld et al. 2017; Momennejad et al. 2018).

When I trained the STDP model on a population of homogeneously-distributed multiscale basis features, the resulting weight matrix displayed binding across the different sizes regardless of the scale difference (Figure 3.4c top). This in turn leads to a population of downstream successor features with the same redundantly large scale (Figure 3.4c bottom). The negative interaction between different sized fields was not sufficient to prevent binding and, as such, the place fields of small features are dominated by contributions

from bindings to larger basis features. Conversely, when these multiscale basis features were ordered along the dorso-ventral axis to prevent binding between the different scales – cells of the three scales were processed separately, Figure 3.4d top) – the multiscale structure is preserved in the resulting successor features (Figure 3.4d bottom). I thus propose that place cell size can act as a proxy for the predictive time horizon, τ – also called the discount parameter, $\gamma = e^{-\frac{dt}{\tau}}$, in discrete Markov Decision Processes. However for this effect to be meaningful, plasticity between cells of different scales must be minimised to prevent short timescales from being overwritten by longer ones, this segregation may plausibly be achieved by the observed size ordering along the hippocampal dorsal-ventral axis.

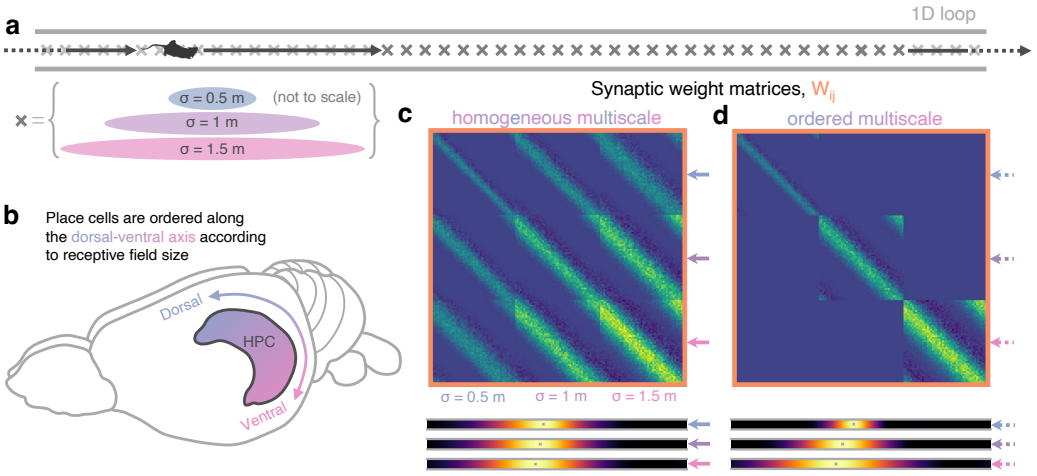

Figure 3.4: Multiscale successor representations are stored by place cells with multi-sized place fields but only when sizes are segregated along the dorso-ventral axis. **a** An agent explores a 1D loop maze with 150 places cells of different sizes (50 small, 50 medium, and 50 large) evenly distributed along the track. **b** In rodent hippocampus place cells are observed to be ordered along the dorso-ventral axis according to their field size. **c** When cells with different field sizes are homogeneously distributed throughout hippocampus all postsynaptic successor features can bind to all presynpatic basis features, regardless of their size (top). Short timescale successor representations are overwritten, creating three equivalent sets of redundantly large scale successor features (bottom). **d** Ordering cells leads to anatomical segregation; postsynaptic successor features can only bind to basis features in the same size range (off-diagonal block elements are zero) preventing cells with different size fields from binding. Now, three dissimilar sets of successor features emerge with different length scales, corresponding to successor features of different discount time horizons.

3.3 Discussion: Bridging Synaptic Plasticity and Reinforcement Learning

Successor representations store long-run transition statistics and allow for rapid prediction of future states (Dayan 1993) - they are hypothesised to play a central role in mammalian navigation strategies (Stachenfeld et al. 2017; Cothi et al. 2020b). It is shown that Hebbian learning between spiking neurons, resembling the place fields found in CA3 and CA1, learns an accurate approximation to the successor representation when these neurons undergo phase precession with respect to the hippocampal theta rhythm. The approximation achieved by STDP explains a large proportion of the variance in the TD successor matrix and replicates hallmarks of successor representations (Stachenfeld et al. 2014; Stachenfeld et al. 2017; Cothi et al. 2020b) such as behaviourally biased place field skewing, elongation of place fields near walls, and clustering near doorways in both one and two-dimensional environments.

That the predictive skew of place fields can be accomplished with a STDP-type learning rule is a long-standing hypothesis; in fact, the authors that originally reported this effect also proposed a STDP-type mechanism for learning these fields (Mehta et al. 2000; Mehta 2001). Similarly, the possible accelerating effect of theta phase precession on sequence learning has also been described in a number of previous works (Jensen et al. 1996; Skaggs et al. 1996b; Koene et al. 2003; Reifenstein et al. 2021). Until recently (Fang et al. 2022; Bono et al. 2021), SR models have largely not connected with this literature: they either remain agnostic to the learning rule or assume temporal difference learning (which has been well-mapped onto striatal mechanisms (Schultz et al. 1997; Seymour et al. 2004), but it is unclear how this is implemented in hippocampus) (Stachenfeld et al. 2014; Stachenfeld et al. 2017; Cothi et al. 2020b; Geerts et al. 2020; Vértes et al. 2019). Thus, one contribution of this chapter is to quantitatively and qualitatively compare theta-augmented STDP to temporal difference learning, and demonstrate where these functionally overlap. This explicit link permits some insights about the physiology, such as the observation that the biologically

observed parameters for phase precession and STDP resemble those that are optimal for learning the SR (Figure C.3), and that the topographic organisation of place cell sizes is useful for learning representations over multiple discount timescales (Figure 3.4). It also permits some insights for RL, such as that the approximate SR learned with theta-augmented STDP, while provably theoretically different from TD (Appendix C.9), is sufficient to capture key qualitative phenomena.

Theta phase precession has a dual effect not only *allowing* learning by compressing trajectories to within STDP timescales but also *accelerating* convergence to a stable representation by arranging the spikes from cells along the current trajectory to arrive in the order those cells are actually encountered (Jensen et al. 1996; Koene et al. 2003). Without theta phase precession, STDP fails to learn a successor representation reflecting the current policy unless that policy is approximately unbiased. Further, by instantiating a population of place cells with multiple scales I show that topographical ordering of these place cells by size along the dorso-ventral hippocampal axis is a necessary feature to prevent small discount timescale successor representations from being overwritten by longer ones. Last, performing a grid search over STDP learning parameters, I show that those values selected by evolution are approximately optimal for learning successor representations. This finding is compatible with the idea that the necessity to rapidly learn predictive maps by STDP has been a primary factor driving the evolution of synaptic learning rules in hippocampus.

While the model is biologically plausible in several respects, there remain a number of aspects of the biology that are not interfaced with in this work, such as different cell types, interneurons and membrane dynamics. Further, I do not consider anything beyond the most simple model of phase precession, which directly results in theta sweeps in lieu of them developing and synchronising across place cells over time (Feng et al. 2015). Rather, the philosophy here is to reconsider the most pressing issues with the standard model of predictive

map learning in the context of hippocampus (e.g., the absence of dopaminergic error signals in CA1 and the inadequacy of synaptic plasticity timescales). This minimalism is believed to be helpful, both for interpreting the results presented here and providing a foundation on which further work may examine these biological intricacies, such as whether the model's theta sweeps can alternately represent future routes (Kay et al. 2020) e.g., by the inclusion of attractor dynamics (Chu et al. 2022). Still, I show this simple model is robust to the observed variation in phase offsets between phase precessing CA3 and CA1 place cells across different stages of the theta cycle (Mizuseki et al. 2012). In particular, this phase offset is most pronounced as animals enter a field ($\sim 90°$) and is almost completely reduced by the time they leave it ($\sim 90°$) (Figure 3.2, figure supplement 4g). Essentially the model hypothesises that the majority of plasticity induced by STDP and theta phase precession will take place in the latter part of place fields, equating to earlier theta phases. Notably, this is in keeping with experimental data showing enhanced coupling between CA3 and CA1 in these early theta phases (Colgin et al. 2009; Hasselmo et al. 2002). However, as the simulations show (Figure 3.2, figure supplement 4g), even if these assumptions do not hold true, the model is sufficiently robust to generate SR equivalent weight matrices for a range of possible phase offsets between CA3 and CA1.

This model extends previous work – which required successor features to recursively expand in order to make long range predictions (e.g. as demonstrated in Brea et al. (2016) and Bono et al. (2021)) – by exploiting the existence of temporally compressed theta sweeps (O'Keefe et al. 1993; Skaggs et al. 1996b), allowing place cells with distant fields to bind directly without intermediaries or 'bootstrapping'. This configuration yields several advantages. First, learning with theta sweeps converges considerably faster than without them. Biologically, it is likely that successor feature learning via Hebbian learning alone (without theta precession) would be too slow to account for the rapid stabilisation of place cells in new environments at behavioural time scales (Bittner et al. 2017) –

Dong et al. observed place fields in CA1 to increase in width for approximately the first 10 laps around a 3 m track (Dong et al. 2021). This timescale is well matched by my model with theta sweeps in which CA1 place cells reach 75% of their final extent after 5 minutes (or 9.6 laps) of exploration on a 5m track but is markedly slower without theta sweeps.

Second, as well as extending previous work to large two-dimensional environments and complex movement policies this model also uses realistic population codes of overlapping Gaussian features. These naturally present a hard problem for models of spiking Hebbian learning since, in the absence of theta sweeps, the order in which features are encountered is not encoded reliably in the relative timing or order of their spikes at synaptic timescales. Theta sweeps address this by tending to sequence spikes according to the order in which their originating fields are encountered. Indeed, preliminary experiments show that when theta sweeps are absent the STDP successor features show little similarity to the TD successor features. This work is thus particularly relevant in light of a recent trend to focus on biologically plausible features for reinforcement learning (Gustafson et al. 2011; Cothi et al. 2020b).

Other contemporary theoretical works have made progress on biological mechanisms for implementing the successor representation algorithm using somewhat different but complementary approaches. Of particular note are the works by Fang et al. Fang et al. (2022), who show a recurrent network with weights trained via a Hebbian-like learning rule converges to the successor representation in steady state, and Bono et al. Bono et al. (2021) who derive a learning rule for a spiking feed-forward network which learns the SR of one-hot features by bootstrapping associations across time (see also Brea et al. (2016)). Combined, the above models, as well as this work suggest there may be multiple means of calculating successor features in biological circuits without requiring a direct implementation of temporal difference learning.

This theory makes the prediction that theta contributes to *learning* predictive representations, but is not necessary to maintain them. Thus,

inhibiting theta oscillations during exposure to a novel environment should impact the formation of successor features (e.g., asymmetric backwards skew of place fields) and subsequent memory-guided navigation. However, inhibiting theta in a familiar environment in which experience-dependent changes have already occurred should have little effect on the place fields: that is, some asymmetric backwards skew of place fields should be intact even with theta oscillations disrupted. To my knowledge this has not been directly measured, but there are some experiments that provide hints. Experimental work has shown that power in the theta band increases upon exposure to novel environments (Cavanagh et al. 2011) – this work suggests this is because theta phase precession is critical for learning and updating stored predictive maps for spatial navigation. Furthermore, it has been shown that place cell firing can remain broadly intact in familiar environments even with theta oscillations disrupted by temporary inactivation or cooling (Bolding et al. 2019; Petersen et al. 2020). It is worth noting, however, that even with intact place fields, these theta disruptions impair the ability of rodents to reach a hidden goal location that had already been learned, suggesting theta oscillations play a role in navigation behaviours even after initial learning (Bolding et al. 2019; Petersen et al. 2020). Other work has also shown that muscimol inactivations to medial septum can disrupt acquisition and retrieval of the memory of a hidden goal location (Chrobak et al. 1989; Rashidy-Pour et al. 1996), although it is worth noting that these papers use muscimol lesions, which Bolding and colleagues show also disrupt place-related firing, not just theta precession.

The SR model has a number of connections to other models from the computational hippocampal literature that bear on the interpretation of these results. A long-standing property of computational models in the hippocampus literature is a factorisation of spatial and reward representations (Redish et al. 1998; Burgess et al. 1997; Koene et al. 2003; Hasselmo et al. 2005; Erdem et al. 2012), which permits spatial navigation to rapidly adapt to changing goal locations. Even in RL, the SR is also not unique in factorising spatial and

reward representations, as purely model-based approaches do this too (Dayan 1993; Sutton et al. 1998; Daw 2012). The SR occupies a much more narrow niche, which is factorising reward from spatial representations while caching long-term occupancy predictions (Dayan 1993; Gershman 2018). Thus, it may be possible to retain some of the flexibility of model-based approaches while retaining the rapid computation of model-free learning.

A number of other models describe how physiological and anatomical properties of hippocampus may produce circuits capable of goal-directed spatial navigation (Erdem et al. 2012; Redish et al. 1998; Koene et al. 2003). These models adopt an approach more characteristic of model-based RL, searching iteratively over possible directions or paths to a goal Erdem et al. (2012) or replaying sequences to build an optimal transition model from which sampled trajectories converge toward a goal Redish et al. (1998) (this model bears some similarities to the SR that are explored by Fang et al. (2022), which shows dynamics converge to SR under a similar form of learning). These models rely on dynamics to compute the optimal trajectory, while the SR realises the statistics of these dynamics in the rate code and can therefore adapt very efficiently. Thus, the SR retains some efficiency benefits. These models are very well-grounded in known properties of hippocampal physiology, including theta precession and STDP, whereas until recently, SR models have enjoyed a much looser affiliation with exact biological mechanisms. Thus, a primary goal of this chapter is to explore how hippocampal physiological properties relate to SR learning as well.

More generally, in principle, any form of sufficiently ordered and compressed trajectory would allow STDP plasticity to approximate a successor representation. Hippocampal replay is a well documented phenomena where previously experienced trajectories are rapidly recapitulated during sharp-wave ripple events (Wilson et al. 1994), within which spikes show a form of phase precession relative to the ripple band oscillation (150-250Hz) (Bush et al. 2022). Thus, my model might explain the abundance of sharp-wave ripples during early

exposure to novel environments (Cheng et al. 2008) – when new 'informative' trajectories, for example those which lead to reward, are experienced it is desirable to rapidly incorporate this information into the existing predictive map (Mattar et al. 2018).

The distribution of place cell receptive field size in hippocampus is not homogeneous. Instead, place field size grows smoothly along the longitudinal axis (from very small in dorsal regions to very large in ventral regions). Why this is the case is not clear – my model contributes by showing that, without this ordering, large and small place cells would all bind via STDP, essentially overwriting the short timescale successor representations learnt by small place cells with long timescale successor representations. Topographically organising place cells by size anatomically segregates place cells with fields of different sizes, preserving the multiscale successor representations. Further, my results exploring the effect of different phase offsets on STDP-successor learning (Figure C.4g) suggest that the gradient of phase offsets observed along the dorso-ventral axis (Lubenov et al. 2009; Patel et al. 2012) is insufficient to impair the plasticity induced by STDP and phase precession. The premise that such separation is needed to learn multiscale successor representations is compatible with other theoretical accounts for this ordering. Specifically Momennejad and Howard Momennejad et al. (2018) showed that exploiting multiscale successor representations downstream, in order to recover information which is 'lost' in the process of compiling state transitions into a single successor representation, typically requires calculating the derivative of the successor representation with respect to the discount parameter. This derivative calculation is significantly easier if the cells – and therefore the successor representations – are ordered smoothly along the hippocampal axis.

Work in control theory has shown that the difficult reinforcement learning problem of finding an optimal policy and value function for a given environment becomes tractable if the policy is constrained to be near a 'default policy' (Todorov 2009). When applied to spatial navigation, the optimal value function

resembles the value function calculated using a successor representation for the default policy. This solution allows for rapid adaptation to changes in the reward structure since the successor matrix is fixed to the default policy and need not be re-learnt even if the optimal policy changes. Building on this, recent work suggested the goal of hippocampus is not to learn the successor representation for the current policy but rather for a default diffusive policy (Piray et al. 2021).

Indeed, I found that in the absence of theta sweeps, the STDP rule learns a successor representation close to that of an unbiased policy, rather than the current policy. This is because without theta-sweeps to order spikes along the current trajectory, cells bind according to how overlapping their receptive fields are, that is, according to how close they are under a 'diffusive' policy. In this context it is interesting to note that a substantial proportion of CA3 place cells do not exhibit significant phase precession (O'Keefe et al. 1993; Jeewajee et al. 2014). One possibility is that these place cells with weak or absent phase precession might plausibly contribute to learning a policy-independent 'default representation', useful for rapid policy prediction when the reward structure of an environment is changed. Simultaneously, theta precessing place cells may learn a successor representation for the current (potentially biased) policy, in total giving the animal access to both an off-policy-but-near-optimal value function and an on-policy-but-suboptimal value function.

Finally, a comment is made on the approximate nature of the successor representations learnt by my biologically plausible model. The STDP successor features described here are unlikely to converge analytically to the TD successor features. Potentially this implies that a value function calculated according to Equation (C.28) would not be accurate and may prevent an agent from acting optimally. There are several possible resolutions to this point. First, the successor representation is unlikely to be a self contained reinforcement learning system. In reality it likely interacts with other model-based or model-free systems acting in other brain regions such as nucleus accumbens in striatum

(Lisman et al. 2005). Plausibly errors in the successor features are corrected for by counteracting adjustments in the reward weights implemented by some downstream model-free error-based learning system. Alternatively, it is likely that value function learnt by the brain is either fundamentally approximate or uses an different, less tractable temporal discounting scheme. Ultimately, although in principle specialised and expensive learning rules might be developed to exactly replicate TD successor features in the brain, this may be undesirable if a simple learning rule (STDP) is adequate in most circumstances. Indeed, animals - including humans - are known to act sub-optimally (Zentall 2015; Cothi et al. 2022b), perhaps in part because of a reliance on STDP learning rules in order to learn long-range associations.

Chapter 4

A Biologically Plausible Generative Model of the Hippocampal Formation

Summary

Advances in generative models have recently revolutionized machine learning. Meanwhile, in neuroscience, generative models have long been thought fundamental to animal intelligence. Understanding the biological mechanisms that support these processes promises to shed light on the relationship between biological and artificial intelligence. In animals, the hippocampal formation is thought to learn and use a generative model to support its role in spatial and non-spatial memory. Here, a biologically plausible model of the hippocampal formation tantamount to a Helmholtz machine is introduced that is applied to a temporal stream of inputs. A novel component of the model is that fast theta-band oscillations (5-10 Hz) gate the direction of information flow throughout the network, training it akin to a high-frequency wake-sleep algorithm. The model accurately infers the latent state of high-dimensional sensory environments and generates realistic sensory predictions. Furthermore, it can learn to path integrate by developing a ring attractor connectivity structure matching previous theoretical proposals and flexibly transfer this structure between

environments. Whereas many models trade-off biological plausibility with generality, the model captures a variety of hippocampal cognitive functions under one biologically plausible local learning rule.

4.1 Introduction: A Theta-Driven Generative Model of the Hippocampus

Generative models seek to create new data samples which are similar to those from the training set. To do so they must learn the probability distribution of the training data, comprising a rich, generalisable and accurate model of the world. Many of the recent advances in AI have involved types of generative models: VAEs (Kingma et al. 2022), GANs (Goodfellow et al. 2014), diffusion models (Sohl-Dickstein et al. 2015) and autoregressive models (Vaswani et al. 2017) have seeded improvements in AI capabilities ranging from data compression (Yang et al. 2022; George et al. 2019) to image generation (Ramesh et al. 2021) and natural language (Bubeck et al. 2023). In neuroscience, the animal brain has long been known to exploit generative models (Friston 2010; Gershman 2019). The ability to generate representative sensory data samples can be used directly, for example during offline planning or memory recall. It can also be used indirectly to aid training of inference networks with the goal of processing rich, noisy and high dimensional streams of incoming sensory stimuli, as discussed in the predictive coding literature (Rao et al. 1999). In a sentence: "What I cannot create [generate], I do not understand [inference]" (R. Feynman).

The hippocampal-entorhinal system (aka. hippocampal formation) – a brain structure implicated in spatial (O'Keefe 1976) and non-spatial (Squire 1992) memory – provides a pertinent example. Its primary role seems to be inference (Sanders et al. 2020): mapping sensory inputs into a robust and decodable representation of state (grid cells (Hafting et al. 2005), place cells (O'Keefe 1976) etc. (Moser et al. 2017)). A generative model is thought to have a dual role in learning: supporting offline tasks such as route planning (Spiers

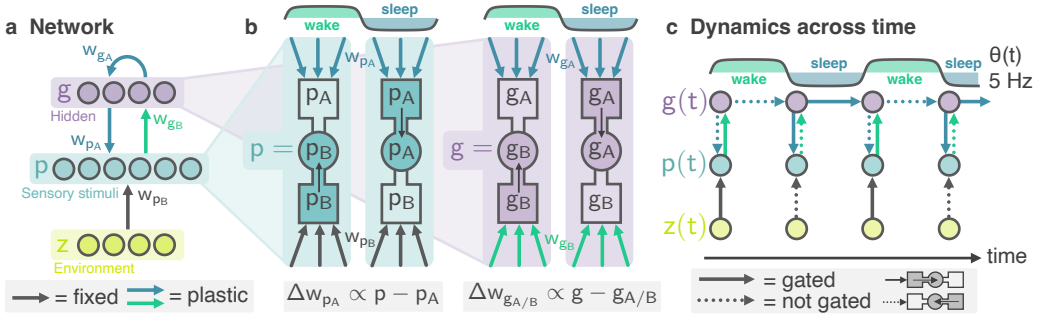

Figure 4.1: A biologically plausible generative model is trained with theta frequency wake-sleep cycles and a local learning rule. **a** Network schematic: high-D stimuli from an underlying environmental latent state z arrive at the basal dendrites of the sensory layer, p, and map to the hidden layer, g (this is the inference model, weights in green). Simultaneously, top-down predictions from the hidden layer g arrive at the apical dendrites of p (this is the generative model, weights in blue). **b** Neurons in layers p and g have three compartments. A fast oscillation, $\theta(t)$, gates which dendritic compartment – basal (p_B, g_B) or apical (p_A, g_A) – drives the soma. A local learning rule adjusts input weights to minimise the prediction error between dendritic compartments and the soma. **c** This equates to rapidly switching "wake" and "sleep" cycles which train the generative and inference models. Panel (c) displays just two updates per theta-cycle; in reality, there are many ($\delta t << T_\theta$).

et al. 2006) and memory consolidation (Carr et al. 2011), and online during behaviour with path integration (McNaughton et al. 1996). Path integration enables the hippocampal network to maintain an up-to-date and accurate estimate of its position in the absence of reliable sensory data by integrating self-motion cues. A recent flurry of computational (Cueva et al. 2018; Banino et al. 2018; Sorscher et al. 2023) and theoretical (Dorrell et al. 2023; Sorscher et al. 2023) work has highlighted the importance of path integration as a key objective explaining hippocampal function and representations.

Existing computational generative models of the hippocampal formation (Whittington et al. 2020; George et al. 2021) account for many of its cognitive functions and internal representations but require non-trivial learning rules and message passing protocols that don't connect with known aspects of biology. Computational models of path integration (Skaggs et al. 1995; Samsonovich et al. 1997; Burak et al. 2009) have mostly focussed on continuous attractor networks which, although experimentally supported (Khona et al. 2021), alone lack the complexity or expressivity required of a fully general model of the

hippocampal memory system.

The primary contribution of this chapter is to introduce a biologically plausible model of sequence learning in the hippocampus which unifies its capacities as a generative model of sensory stimuli and path integration under one schema. To do this I propose modelling the hippocampal formation as a Helmholtz machine (Dayan et al. 1995) which learns to predict sensory stimuli given the current hidden state and action (e.g. velocity). I propose a deep connection between the hippocampal theta oscillation (Buzsáki 2002) and the unsupervised wake-sleep algorithm (Hinton et al. 1995) for training Helmholtz machines. Though this class of generative models isn't widely used, and lacks the scalability of the latest transformer-based sequence learners, it excels in this context since it has many natural points of contact with biology (both in terms of architecture and neural dynamics) yet still maintains the expressiveness afforded to models of the brain by deep neural networks.

In this chapter I:

- introduce a new model of the hippocampal formation which learns the latent structure of an incoming stream of sensory stimuli analogous to a Helmholtz machine.

- describe a biologically plausible learning regime: Theta-oscillations gate information flow through multi-compartmental neurons which rapidly switches the system between "wake" and "sleep" phases. All plasticity is local.

- train the model on stimuli from a biologically relevant spatial exploration task and show it learns to path integrate by developing a ring attractor connectivity structure (comparable to theoretical predictions and empirical results in deep recurrent neural networks trained with gradient descent). Learning generalises: when the agent moves to a new environment, path integration capabilities recover without needing to relearn the path integration weights.

This model of the hippocampal formation simultaneously (i) accounts for its role as a generative model of sensory stimuli, (ii) can learn to path integrate and (iii) can transfer structural knowledge between environments. The model, though here applied to the hippocampus, can be viewed as a step towards a general solution for how biological neural networks in many brain regions (for example visual cortex (Rao et al. 1999)) can learn generative models of the world.

4.1.1 Related work

A recent generative model of the hippocampus, the Tolman-Eichenbaum Machine (Whittington et al. 2020), proposed that the hippocampal formation be thought of as a hierarchical network performing latent state inference. Medial entorhinal cortex (MEC) sits atop the hierarchy and learns an abstract representation of space which is mapped to the hippocampus (HPC) where it is bound onto incoming sensory stimuli. Once trained the system can act in a generative fashion by updating the hidden representation with idiothetic action signals and then predicting the upcoming sensory experience. The drawback of this model, and others which share a similar philosophical approach (Uria et al. 2020; George et al. 2021), is that it requires training via backpropagation through time (or equivalent end-to-end optimisation schemes, as in (George et al. 2021)) without clear biological correlates. Related hierarchical network architectures have also been studied in the context of reinforcement learning (Han et al. 2020) and hippocampal associative memory (Sharma et al. 2022).

Historically, hippocampal models of path integration have focused on continuous attractor networks (CANs) (Skaggs et al. 1995; Samsonovich et al. 1997; Burak et al. 2009; Sorscher et al. 2023) in entorhinal cortex. A bump of activity representing location is pushed around the CAN by speed and/or head-direction selective inputs, thus integrating self-motion. CANs have received substantial experimental support (Khona et al. 2021) but few studies adequately account for *how* this structure is learned by the brain in the first place. One exception exists outside the hippocampal literature: Vafidis et al. (2022) built

a model of path integration in the fly head-direction system which uses local learning rules. This work goes further by embedding the path integrator inside a hierarchical generative model. Doing so additionally relaxes the assumption (made by Vafidis et al. (2022) and others (Widloski et al. 2014)) that sensory inputs into the path integrator are predefined and fixed. Instead, by allowing all incoming and outgoing synapses to be learned from random initialisations, I achieve a more generalisable model capable of transferring structure between environments (see section 4.3.3).

Hippocampal theta oscillations have been linked to predictive sequence learning before (Skaggs et al. 1996b; Mehta et al. 2000; George et al. 2023a; George 2023) where research has focused on the compressive effects of theta *sequences* and how these interplay with short timescale synaptic plasticity. Instead of compression, here it is hypothesized that the role of theta is to control the direction information flows through the hierarchical network.

Finally, a recent theoretical work by Bredenberg et al. (2021) derived, starting from principles of Bayesian variational inference, a biologically plausible learning algorithm for approximate Bayesian inference of a hierarchical network model built from multi-compartmental neurons and trained with local learning rules using wake-sleep cycles. Here I build a similar network to theirs (i) extending it to a spatial exploration task and mapping the hidden layers onto those in the hippocampal formation, (ii) simplifying the learning rules and relaxing a discrete-time assumption – instead, opting for a temporally continuous formulation more applicable to biological tasks such as navigation – and (iii) adapting the hidden layer to allow idiothetic action signals to guide updates (aka. path integration). Their work provides a theoretical foundation for the present work, helping to explain *why* learning converges on accurate generative models.

4.2 A Biologically Plausible Generative Model with Theta-Driven Local Learning

In sections 4.2 and 4.3, concise, intuitive descriptions of the model and experiments are given; expanded details can be found in Appendix D.2.

4.2.1 Basic Model Summary

I consider learning in an environment defined by a latent state, $z(t)$, which updates according to stochastic dynamics initially unknown to the network,

$$\frac{dz}{dt} = f_z(t). \tag{4.1}$$

These dynamics depend on the task; first, $z(t)$ is considered to be a set of mutually independent random variables and later, the more realistic task of an agent moving on a 1D track is considered.

The network receives sensory input which is a function of the latent state into a sensory layer, $\mathbf{p}(t)$, and communicates this to a hidden layer (aka "internal state"), $\mathbf{g}(t)$. The network contains both an *inference* (aka. *recognition*) model which infers the hidden state from the sensory input (green arrows, Figure 4.1a) and a *generative* model which updates the hidden state with recurrent synapses and maps this back to the sensory layer (blue arrows). As will soon be identified, these processes correspond to Basal and Apical dendritic compartments of pyramidal neurons, so activations sampled from the inference model are labelled with the subscript B and those from the generative model with the subscript A.[1] In summary

$$\left.\begin{aligned} \mathbf{p}_B(t+\delta t) &= \tilde{\mathbf{p}}(z(t)) \\ \mathbf{g}_B(t+\delta t) &= \sigma_{g_B}(\mathbf{w}_{g_B}\mathbf{p}(t)) \end{aligned}\right\} \quad \text{Inference model} \tag{4.2}$$

[1] These labellings conveniently match the notion that inferences are made from layers Below in the sensory hierarchy (bottom-up) whereas generative predictions arrive from Above (top-down).

$$\left.\begin{aligned}\mathbf{g}_A(t+\delta t) &= \sigma_{g_A}(\mathbf{w}_{g_A}\mathbf{g}(t))\\ \mathbf{p}_A(t+\delta t) &= \sigma_{p_A}(\mathbf{w}_{p_A}\mathbf{g}(t))\end{aligned}\right\} \quad \text{Generative model.} \tag{4.3}$$

$\mathbf{w}_{g_B}, \mathbf{w}_{p_A}, \mathbf{w}_{g_B}$ are matrices of randomly initialised and plastic synaptic weights. $\bar{\mathbf{p}}$ maps the environmental latent into a vector of neural inputs. σ's denote activation functions applied to the dendritic pre-activations – either the identity ($\sigma(x) = x$) or rectified tanh functions ($\sigma(x) = \max(0, tanh(x))$). A small amount of noise is added to the dendritic activations to simulate realistic biological learning.

I believe that the widely adopted convention of modelling neurons as single-compartment perceptrons is limiting. By considering, in a minimal extension, the distributed dendritic structure of real neurons, significant potential for explaining hippocampal learning can be tapped into. Theoretical (Kording et al. 2001; Urbanczik et al. 2014; Sacramento et al. 2018; Richards et al. 2019) and experimental (Bittner et al. 2015; Brankack et al. 1993; Mizuseki et al. 2009) research into credit assignment in biological neurons has identified different roles for basal and apical dendrites: basal dendrites are thought to receive bottom-up drive from sensory inputs whereas apical dendrites receive top-down drive from higher layers in the sensory hierarchy (Larkum 2022). Following this line of research — and matching an equivalent theoretical model of latent state inference described by (Bredenberg et al. 2021) — I identify the inference process with synaptic inputs into a basal dendritic compartment of pyramidal neurons and the generative process with synaptic inputs into an apical dendritic compartment. In summary, each $\mathbf{p}$ and $\mathbf{g}$ neuron in my model has three compartments: a somatic compartment, a basal dendritic compartment and an apical dendritic compartment (Figure 4.1b). Only the somatic activation is used for communication between layers (right hand side of eqs. (4.2) and (4.3)) while dendritic compartment activations are variables affecting internal neuronal dynamics and learning as described below (eqs. (4.4) and (4.6)).

4.2.2 Theta-Gating Information Flow

The dynamics of the somatic activations $\mathbf{p}(t)$ and $\mathbf{g}(t)$ are as follows: the voltage in each soma is either equal to the voltage in the basal compartment *or* the voltage in the apical compartment depending on the phase of an underlying theta oscillation. This is achieved by a simple theta-gating mechanism (Figure 4.1b):

$$\begin{aligned}\mathbf{p}(t) &= \theta(t)\mathbf{p}_B(t) + (1-\theta(t))\mathbf{p}_A(t) \\ \mathbf{g}(t) &= \theta(t)\mathbf{g}_B(t) + (1-\theta(t))\mathbf{g}_A(t).\end{aligned} \tag{4.4}$$

where $\theta(t)$ is a 5 Hz global theta oscillation variable defined by the square wave function:

$$\theta(t) = \begin{cases} 1, & \text{if } t/T \mod 1 \leq 0.5 \\ 0, & \text{if } t/T \mod 1 > 0.5 \end{cases} \tag{4.5}$$

for $T = 1/f_\theta$ and $f_\theta = 5$ Hz, matching the hippocampal theta frequency (5-10 Hz) (Foster et al. 2007). According to this model theta-band oscillations in the hippocampal local field potential gate which dendritic compartment drives the soma. Experimental (Brankack et al. 1993; Holscher et al. 1997; Yamaguchi et al. 2002) and modelling work (Hasselmo et al. 2002) gives provisional support for this assumption.

These local theta-dynamics have global consequences: the early phase ($\theta(t) = 1$) of each theta cycle can be thought of as a "wake" phase where information flows upwards through the network from the environment to the hidden layer, sampling the inference model. The latter phase ($\theta(t) = 0$) of each theta cycle is a "sleep" phase where information flows down from the hidden layer to the sensory units, sampling the generative model. These dynamics are displayed in Figure 4.1.

4.2.3 Hebbian-Style Learning Rules

In contrast to comparable models which are optimised end-to-end using backpropagation through time my model learns synaptic weights according to a local plasticity rule which is a simplified variant of a rule proposed by

Urbanczik et al. (2014). Incoming synaptic projections are continually adjusted in order to minimize the discrepancy between the somatic activation and the dendritic activation. The full learning rules are described in the Appendix D.2 but simplified versions are given here:

$$\begin{aligned}\frac{d\mathbf{w}_{g_B}}{dt} &\propto (\mathbf{g}(t) - \mathbf{g}_B(t))\mathbf{p}(t)^\mathsf{T} \\ \frac{d\mathbf{w}_{p_A}}{dt} &\propto (\mathbf{p}(t) - \mathbf{p}_A(t))\mathbf{g}(t)^\mathsf{T} \\ \frac{d\mathbf{w}_{g_A}}{dt} &\propto (\mathbf{g}(t) - \mathbf{g}_A(t))\mathbf{g}(t)^\mathsf{T}\end{aligned} \tag{4.6}$$

Notably this learning rule is equivalent for *all* plastic synapses in the model: $\mathbf{p}$ to $\mathbf{g}$, $\mathbf{g}$ to $\mathbf{p}$ and the recurrent $\mathbf{g}$ to $\mathbf{g}$ synapses (see Figure 4.1b). If a local prediction error is detected, for example the somatic activation is larger than the dendritic activation, then the synaptic strength of inputs into that dendritic compartment which are positive/negative are strengthened/weakened to reduce the error. This model can equivalently be viewed as a type of Hebbian learning – weight change is proportional to the correlation of pre- and post-synaptic activity (the first term) – regularised (by the second term) to prevent unbounded growth.

During the wake phase the weights of the generative model ($\mathbf{w}_{p_A}$ and $\mathbf{w}_{g_A}$) are trained and plasticity on the inference weights ($\mathbf{w}_{g_B}$) falls to zero. This occurs naturally because $\mathbf{p} = \mathbf{p}_B$ so there will be no basal prediction errors to correct. During sleep the reverse occurs; the weights of the inference model are trained and plasticity on the generative model falls to zero. Experimentally, apical activity is known to guide plasticity at basal synapses in CA1 (Bittner et al. 2015). This alternating, coordinated regime of sampling and learning (sample-inference-train-generative, then sample-generative-train-inference) is a hallmark of the wake-sleep algorithm. It fundamentally differs from the forward and backward sweeps of backpropagation since neurons remain provisionally active at all times so the process of learning minimally perturbs perception. Also, whereas backpropagation sends error signals down through the network

to train synaptic weights, here only predictions are sent between layers and error signals are calculated locally at each dendrite.

As discussed in section 4.1, Bredenberg et al. (2021) mathematically derive learning rules similar to these starting from a loss function closely related to the evidence lower bound (ELBO). As such my identification of early- and late-theta phases as "wake" and "sleep" cycles can be considered precise: from a Bayesian perspective, the hippocampal model is minimising a modified ELBO loss (see Appendix D.3) thus learns to find approximately optimal inference and generative models accounting for the temporally varying stimulus stream it is presented.

4.2.4 Velocity Inputs

For path integration, the hidden state needs access to an idiothetic (internally generated) velocity signal. To satisfy this, the hidden layer, $\mathbf{g}$, is endowed with conjunctive velocity inputs, henceforth "conjunctive cells", as shown in Figure 4.3a,b. Conjunctive cells are organised into two groups: $\mathbf{g}_{v_L}$ is responsible for leftward motion and $\mathbf{g}_{v_R}$ for rightward motion. Each conjunctive cell receives input from the hidden units and either the leftward ($v_L = \max(0, -\dot{x})$) or rightward ($v_R = \max(0, \dot{x})$) component of the velocity. For the results shown this connectivity is one-to-one $[\mathbf{w}_{g_{vL}}]_{ij} = [\mathbf{w}_{g_{vR}}]_{ij} = \delta_{ij}$ but random connectivity works too, see Appendix D.5.2. Finally, conjunctive cells send return connections back to the apical dendritic compartment of the hidden units via a randomly initialised plastic synaptic weight matrix. These inputs are what drive the hidden units to path integrate.

This model takes inspiration from so-called conjunctive grid cells (Sargolini et al. 2006) found in the medial entorhinal cortex (MEC). These cells, thought to be an integral component of the mammalian path integration system (Burak et al. 2009), are jointly tuned to head direction and location much like the conjunctive cells in my model. An important and novel aspect of my model is that synaptic weights between or into the hidden units are *learned*. This deviates from other models, for example, that by Burak et al. (2009) (where all

connectivity is predefined and fixed) or Vafidis et al. (2022) and Widloski et al. (2014) (where sensory inputs to the hidden units are pre-defined and fixed). This is not only more realistic but affords the model flexibility to translate path integration abilities between environments without having to relearn them, a form of transfer learning which is demonstrated in section 4.3.3.

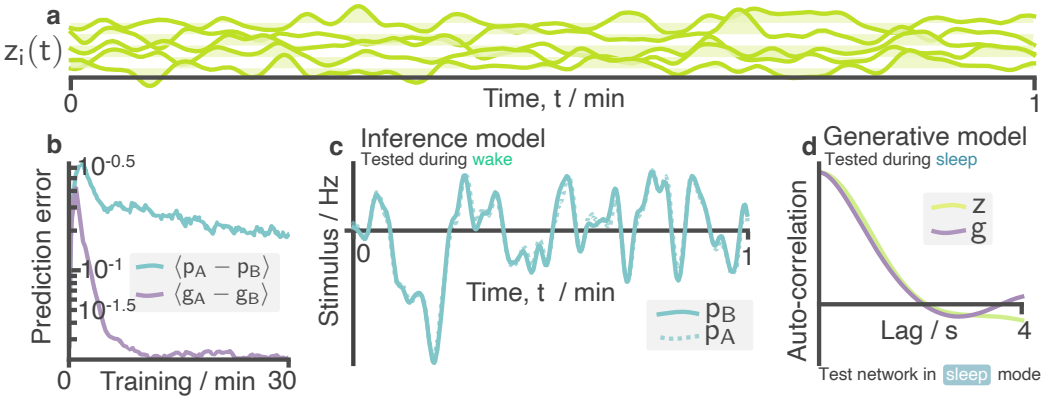

Figure 4.2: Learning in an environment of temporally varying latents. **a** In this artificial task the latent space comprises of $N_z = 5$ independent random variables with an autocorrelation decay timescale of 1 s. **b** Prediction errors (difference between apical and basal activations) in sensory and hidden layers reduce over training time. **c** Tested in wake mode ($\theta = 1$) after training, the ground truth stimulus matches apical prediction for all stimulus dimensions (one shown) implying the network is efficiently "autoencoding" the sensory inputs into and back out of the compressed hidden layer. **d** Tested in sleep mode ($\theta = 0$, no environmental inputs), generated data from the hidden units, g, have an autocorrelation curve which matches that of the true latents implying a statistically accurate generative model has been learned. More extensive samples from this model, before and after training, can be found in Figure D.1

4.3 Model Validation: From Latent Inference to Path Integration

4.3.1 Artificial Latent Task Validation

Testing begins with the basic model (i.e. without conjunctive inputs, Figure 4.1a) on an artificial task. $N_z = 5$ latents, $z_i(t)$, are independently sampled from a smooth, random process with an autocorrelation timescale of 1 second (Figure 4.2a). The sensory layer, $N_p = 50$, then receives a high-

dimensional random linear mixture of the latents into the basal compartments:

$$\mathbf{p}_B(t) = \mathbf{A}\mathbf{z}(t), \tag{4.7}$$

where $\mathbf{A} \in \mathbb{R}^{50\times5}$ and $[\mathbf{A}]_{ij} \sim \mathcal{N}(0, \frac{1}{\sqrt{N_z}})$. The hidden layer, $\mathbf{g}(t)$, is matched in size to the latent process, $N_g = N_z = 5$, and all dendritic activation functions are linear. The model is trained for 30 minutes of simulated time and track prediction errors, the difference between the basal and apical activations in the sensory and hidden layers, which reliably decreased throughout training (Figure 4.2b). Two tests are then performed designed to confirm whether the model has learnt accurate inference and generative models.

First, the dynamics of the model are set to "wake" mode ($\theta = 1$) and the basal and apical activations of one of the sensory neurons are measured for 60 seconds. Close correspondence (Figure 4.2c) confirms that the network accurately "autoencodes" the high-dimensional sensory inputs through the compressed hidden layer. Since all activation functions are linear this implies that $\mathbf{w}_{g_B}$ and $\mathbf{w}_{p_A}$ are pseudoinverses. Next, the network is placed in "sleep" mode ($\theta = 0$) and the generative model is allowed to run freely. The autocorrelation of the generated hidden states ($\mathbf{g}(t|\theta = 0)$, displayed fully in Figure D.1) match that of the true environmental latents ($\mathbf{z}(t)$), Figure 4.2d, implying the generative model has statistics closely matching those of the true underlying generative process.

4.3.2 Emergence of a Ring Attractor for Path Integration

Next I turn my attention to the hippocampal formation's role in spatial navigation, and our central result. The environment consists of an agent randomly moving around a 1 m 1D circular track (motion and cell data is generated using the RatInABox package (George et al. 2024)). The basal compartment of each HPC neuron is spatially tuned to a single different Gaussian input however non-Gaussian randomly spatially tuned inputs work

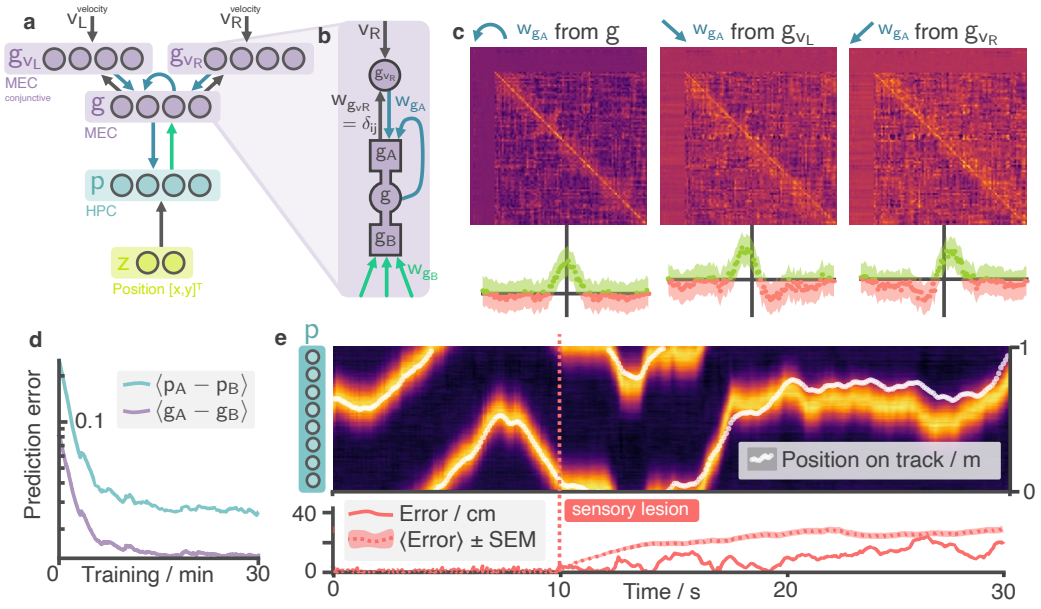

Figure 4.3: The hippocampal model learns to path integrate on a 1D track using a ring attractor. **a** Position selective (place cell) inputs drive basal dendrites of the sensory layer **p** (HPC). **b** Hidden units (MEC) are connected to two sets of "conjunctive cells" which each connect back to one of the hidden neurons (**g**) and either the leftward (for $\mathbf{g}_{v_L}$) or rightward (for $\mathbf{g}_{v_L}$) velocity of the agent allowing velocity information to enter the network. Synaptic strengths of the return connections from the conjunctive cells to the MEC hidden units, as well as those for the MEC recurrent connectivity (collective denoted $\mathbf{w}_{g_A}$), are randomly initialised and plastic. **c** After training, reordering the hidden units by the position of peak activity reveals a ring attractor in the synaptic weight matrices. Centre-surround recurrent connectivity stabilises an activity bump which is then "pushed" around the attractor manifold by asymmetric connections from the conjunctive cells, integrating velocity. Bands of zero weights show MEC neurons which have become perpetually inactive (aka "died"). The bottom panel displays the matrix row-averages, utilizing the circular symmetry of the environment to align rows before averaging. **d** Learning plateaus after 15 mins of simulated time. **e** Path integration ability is demonstrated in a lesion study: after 10 seconds in the normal oscillatory mode the network is placed into sleep mode (aka generative mode), lesioning the position-dependent sensory inputs. Despite this HPC continues to accurately encode position, evidence that the MEC ring attractor is path integrating the velocity inputs and sending predictions back to HPC. Lower panel shows the accumulated decoding error as well as the mean±SEM over 50 trials.

as well (see Figure D.2b):

$$[\mathbf{p}_B(t)]_i = \exp\left[-\frac{(x(t)-x_i)}{2\sigma^2}\right]. \tag{4.8}$$

$x(t)$ is the position of the agent and $\{x_i\}_{i=1}^{N_p}$ are the centres of the Gaussian inputs ($\sigma = 6$ cm), intended to simulate hippocampal place fields, evenly spaced at 1 cm intervals along the track. MEC (i.e. the hidden layer, $\mathbf{g}(t)$) is matched in size $N_g = N_p = 100$ with rectified tanh activation functions on both dendritic compartments ($\sigma_{g_B}(x) = \sigma_{g_A}(x) = \max(0, \tanh(x))$) and HPC (the sensory layer $\mathbf{p}(t)$) is linear ($\sigma_{p_A}(x) = x$). Two populations of conjunctive cells (Figure 4.3a,b) feed into the apical compartments of the MEC recurrent units. Random initialisation of $\mathbf{w}_{g_B}$ means that MEC neurons start off with random non-Gaussian spatial tunings. $\mathbf{w}_{g_A}$ and $\mathbf{w}_{p_A}$ are also randomly initialised.

The network is trained for 30 minutes with learning plateauing after 15 (Figure 4.3d). A lesion study, designed to test path integration, is then performed as follows: First, the network is run for 10 seconds normally (i.e. with theta-oscillating periods of wake and sleep). Since the simulated HPC neurons receive place-tuned inputs uniformly ordered along the track (i.e. $x_j > x_i \forall i, j > i$) an activity heatmap of HPC reveals a bump of activity accurately tracking agent's position (Figure 4.3e, left). The network is then placed into a sleep phase ($\theta = 0$) for 20 seconds. This amounts to a full sensory lesion since top-down MEC inputs, not bottom-up place-tuned sensory inputs, drive HPC. Despite the full sensory lesion, hippocampal activity remains approximately unperturbed and the activity bump continues to accurately track position, slowly accumulating errors (Figure 4.3e right). Since my HPC layer has no recurrent connectivity it cannot support this post-lesion activity on its own. Instead feed-forward drive from an MEC ring attractor, which I turn my attention to now, is responsible for maintaining the HPC code.

To find the ring attractor, the MEC cells must first be reordered. This is done according to the position of the peak of their receptive fields (defined in

Appendix D.5). After reordering, the recurrent connectivity matrix can be seen to have acquired a centre-surround connectivity profile. Nearby MEC cells were, on average, strongly and positively recurrently connected to one another. Those far apart weakly inhibit one another (Figure 4.3c, left; band of strong positive weights along diagonal flanked by weak negative weights). This profile matches that of a quasi-continuous ring attractor: local excitatory and long-range inhibitory connections stabilise a bump of activity on the attractor manifold in the absence of sensory input (Zhang 1996). Weights from the conjunctive cells acquired asymmetric connectivity (Figure 4.3c, middle & right) skewed towards the velocity direction for which they are selective. These asymmetric connections enable conjunctive cells to "push" the activity bump around the manifold, integrating velocity (see Figure D.2 for a visualisation of the MEC bump attractor). Theoretical work on ring attractors has demonstrated that for accurate path integration the asymmetric weights must be proportional to the derivative of the symmetric weights (Zhang 1996), approximately observed here. A noteworthy observation is that some MEC neurons become perpetually inactive; this is a consequence of the fact that *both* top-down and bottom-up synapses into the hidden layer are plastic and can fall to zero (Figure 4.3c bands of zero-weights) satisfying a trivial $g_A = g_B = 0$ solution for minimising the prediction error. Despite this, not all MEC neurons die and the surviving subset are sufficient for path integration. In Appendix D.5.2, additional results are discussed showing the network learns robust path integration under a variety of plasticity, initialisation and noise manipulations.

Crucially, what sets this model apart from others (Cueva et al. 2018; Banino et al. 2018; Sorscher et al. 2023; Dorrell et al. 2023) is that the network is not optimized using a conventional path-integration objective and backpropagation. Instead, it has been demonstrated how path integration can naturally arise in a biologically constrained network subject to a much simpler (yet more broadly applicable) local objective, in cases where idiothetic velocity signals are available to the hidden layers.

4.3.3 Remapping and Flexible Transfer of the Path Integration Circuit

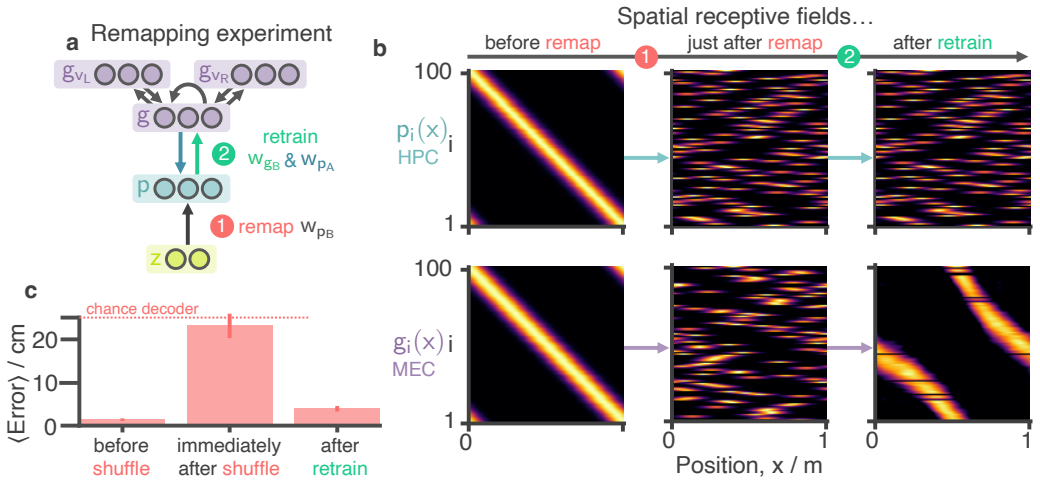

Figure 4.4: Remapping and transfer of structural knowledge between environments. **a** After training (as in Figure 4.2) place cell inputs are shuffled to simulate a "remapping" event observed when an agent moves to a new environment. The agent then retrains for an additional 30 minutes: during this period internal MEC weights, and weights from the conjunctive cells to MEC are held fixed while MEC ↔ HPC weights remain plastic. **b** Receptive fields of the HPC and MEC neuronal populations at different stages in the experiment: Initially after remapping HPC and MEC inputs are randomised. MEC relearns rate maps as they were before remapping but with a constant phase shift. Note: neurons are ordered by the position of their peak activity on the track *before* remapping and this ordering is maintained in subsequent panels. **c** The error (± SEM over 50 trials) after 1 second of path integration is shown at different stages of the experiment. Although path integration is initially disrupted after remapping it recovers despite no relearning of the MEC synapses where the ring attractor is stored.

Finally, it is demonstrated how the trained network can transfer structural knowledge – which here means the ring attractor and thereby path integration – between environments. The process starts by training the network as in section 4.3.2; the only difference is that for simplicity I choose to fix $\mathbf{w}_{g_B} = \delta_{ij}$ giving rise to MEC representations which, like HPC, are unimodal (this constraint can be relaxed and, in the more general case, MEC units typically have multiple receptive fields, Figure D.4d, reminiscent of grid cells). I then simulate a hippocampal "remapping" event by shuffling the sensory inputs to the HPC layer (Figure 4.4a,b, top panel) and retraining the network for

a further 30 minutes but this time holding weights in the hidden layer, $\mathbf{w}_{g_A}$. Only the HPC $\leftrightarrow$ MEC synapses ($\mathbf{w}_{g_B}$ & $\mathbf{w}_{p_A}$) remain plastic during retraining. Biologically this may be accounted for by the observation that cortical plasticity is substantially slower than hippocampal plasticity (Ergorul et al. 2006).

During biological remapping events place cells remap independently whereas grid cells remap *en masse* with entire modules shifting by the same constant phase (Fyhn et al. 2007). This observation is reproduced in my model: after retraining MEC units regroup with receptive fields as they were before remapping but with a constant phase shift along the track. This re-emergence of structure occurs because the ring attractor seeds a bump of activity on the attractor manifold (during the "sleep" phases of retraining) onto which the shuffled HPC inputs then bind. Since nothing constrains *where* on the circularly symmetric attractor manifold this regrouping can initiate, only relative correlations, modulo a phase shift, are preserved.

Decoding error one second after a sensory lesion is tested just *before* remapping, just *after* remapping and after retraining (Figure 4.4c). After the remapping path integration abilities temporarily disappear because the MEC ring attractor is still tuned to the old and invalid HPC receptive fields. After relearning – and despite *no adjustments to the MEC weights,* $\mathbf{w}_{g_A}$*, where the ring attractor is stored* – path integration recovers to almost the level before remapping. This differs substantially from other local models of path integration learning (Vafidis et al. 2022; Widloski et al. 2014) which don't consider plasticity on the ring attractor inputs. In these models, adaptation to a new environment necessarily requires complete relearning of the ring attractor. Instead my model exploits the basic fact that movement (path integration) in one environment is fundamentally the same as in another, one must simply learn a new mapping to/from the ring attractor, "translating" it to fit the new sensory stimuli.

4.4 Discussion: Theta Oscillations as a Learning Scheduler for Generative Computation

I propose that the hippocampal formation resembles a Helmholtz machine, simultaneously learning an inference and generative model of sensory stimuli. Like previous models (Whittington et al. 2020) medial entorhinal cortex (MEC) sits hierarchically above the hippocampus (HPC) to which it sends generative predictions. The model differs in the learning rules and neural dynamics: local prediction errors are minimised between distinct dendritic compartments receiving bottom-up and top-down signals. Theta oscillations regulate internal neural dynamics, switching the network between wake and sleep phases. In a navigation task, the MEC model forms a ring attractor capable of path integration. Despite simple learning rules and dynamics, the model retains key cognitive capabilities of the hippocampal formation including the ability to transfer knowledge across different sensory environments.

Local learning rules are commonly recognised as essential in biologically plausible learning algorithms (Urbanczik et al. 2014). However, the importance of learning *scheduling* – how neural systems coordinate or multiplex distinct phases of forward and backward information flow – is often overlooked (Guerguiev et al. 2017). Neural oscillations such as theta, hypothesized to temporally coordinate communication between neuronal populations (Fries 2015), likely play an underexplored role in this regard (neural "bursting" has also been pointed out as a potential solution to multiplexing (Payeur et al. 2021)). One advantage of the wake-sleep algorithm, which this study suggests neural oscillations can support, compared to forward and backward sweeps is that, during convergence, the two phases become highly similar, allowing learning to proceed without affecting perception.

While the discussion has primarily focused on theta oscillations as a mechanism for learning, they have also been proposed as a mechanism for short-range future prediction via so-called "mind-travel" (Sanders et al. 2015). During the latter phase of each theta cycle (i.e. the sleep phase) gain amplified velocity

signals might rapidly drive the MEC activity bump along the manifold allowing the agent to assess nearby upcoming locations. This complimentary proposition could neatly integrate into the framework proposed here and emphasizes the need for further investigation into the multifaceted functions of neural rhythms within the hippocampal/entorhinal system.

Beyond theta oscillations, both faster gamma cycles (Li et al. 2021) and the slower physiological states of sleep and wake (Skaggs et al. 1996a) have been associated with learning. Based on the model I suggest a tentative hypothesis that theta oscillations may be favored due to an optimality criterion; whilst faster oscillations could be a mechanism to prevent extreme drift during sleep that might disrupt learning their frequency might by upper bounded biophysically by the neural time constants associated with the biophysical processes supporting dendritic gating the soma. These ideas, their relevance to other brain regions involved in generative learning, 2D spatial dynamics, and offline memory consolidation/replay remain exciting questions for future theoretical and experimental investigation.

Chapter 5

SIMPL: A Neural Latent Variable Model

Summary

Neural activity in the brain is known to encode low-dimensional, time-evolving, behaviour-related variables. A long-standing goal of neural data analysis has been to identify these variables and their mapping to neural activity. A productive and canonical approach has been to simply visualise neural "tuning curves" as a function of behaviour. In reality, significant discrepancies between behaviour and the true latent variables, such as an agent thinking of position Y whilst located at position X, distort and blur the tuning curves, decreasing their interpretability. To address this, latent variable models propose to learn the latent variable from data; these are typically expensive, hard to tune, or scale poorly, complicating their adoption. Here I propose SIMPL (Scalable Iterative Maximization of Population-coded Latents), an EM-style algorithm which iteratively optimises latent variables and tuning curves. SIMPL is fast, scalable and exploits behaviour as an initial condition to further improve convergence and identifiability. It can accurately recover latent variables in spatial and non-spatial tasks. When applied to a large hippocampal dataset, SIMPL converges on smaller, more numerous, and more uniformly sized place fields than those based on behaviour, suggesting the brain may encode space

with greater resolution than previously thought.

5.1 Introduction: The Discrepancy Between Behavior and Internal State

Large neural populations in the brain are known to encode low-dimensional, time-evolving latent variables which are, oftentimes, closely related to behaviour (Afshar et al. 2011; Harvey et al. 2012; Mante et al. 2013; Carnevale et al. 2015). Coupled with the advent of modern neural recording techniques (Jun et al. 2017; Wilt et al. 2009) focus has shifted from single-cell studies to the joint analysis of hundreds of neurons across long time windows, where the goal is to extract latents using a variety of statistical (Yu et al. 2008a; Cunningham et al. 2014; Kobak et al. 2016; Zhao et al. 2017; Williams et al. 2020; Bjerke et al. 2023) and computational (Maaten et al. 2008; Pandarinath et al. 2018; Mackevicius et al. 2019) methods.

This paradigm shift is particularly pertinent in mammalian spatial memory and motor systems where celebrated discoveries have identified cells whose neural activity depends on behavioural variables such as position (O'Keefe et al. 1971; Hafting et al. 2005), heading direction (Taube et al. 1990), speed (McNaughton et al. 1983), distance to environmental boundaries/objects (Lever et al. 2009; Høydal et al. 2019) and limb movement direction (Georgopoulos et al. 1986) through complex and non-linear tuning curves. Characterising neural activity in terms of behaviour remains a cornerstone practice in these fields however the implicit assumption supporting it — that the latent variable encoded by neural activity *is and only is* the behavioural variable — is increasingly being called into question (Sanders et al. 2015; Whittington et al. 2020; George et al. 2023b).

The brain is not a passive observer of the world. The same neurons that encode an animal's *current* position/behavioural state are also used to plan future routes (Spiers et al. 2006), predict upcoming states (Muller et al. 1989; Mehta et al. 1997; Stachenfeld et al. 2017) or recall/"replay" past positions

(Squire et al. 2010; Carr et al. 2011), necessarily causing the encoded latent variables to deviate from behaviour. Nor is the brain a perfect observer; uncertainty due to limited, noisy or ambiguous sensory data can lead to similar discrepancies. Measurement inaccuracies can contribute further. These hypotheses are supported by analyses that show that it is rarely, if ever, possible to perfectly decode "behaviour" from neural data (Glaser et al. 2020) and the observation that neurons show high variability under identical behavioural conditions (Fenton et al. 1998; Low et al. 2018). All combined, these facts hint at a richer and more complex internal neural code. When this is not accounted for tuning curves will be blurred, distorted or mischaracterised relative to their true form. For example, consider an animal situated at position X 'imagining' or 'anticipating' a remote position, Y, for which a place cell is tuned. This might trigger the cell to fire leading to the mistaken conclusion that the cell has a place field at location X.

Nonetheless, the fact that behaviour is often a close-but-imperfect proxy for the true latent motivates searching for techniques that *exploit* this link. Most existing methods for latent discovery don't exploit behaviour (Gao et al. 2016; Gondur et al. 2023) at the cost of complexity and interpretability. Others don't model temporal dynamics (Zhou et al. 2020; Schneider et al. 2023; Lawrence 2003), don't scale to large datasets (Wang et al. 2005; Nam 2015; Wu et al. 2017), can't model complex non-linear tuning curves (Pandarinath et al. 2018; Hurwitz et al. 2021; Duncker et al. 2019; Linderman et al. 2016; Gondur et al. 2023), or aren't designed for spiking datasets (Lawrence 2003; Krishnan et al. 2015). Moreover, many of these methods are conceptually complex, lack usable code implementations, or necessitate GPUs limiting their accessibility.

Contributions Here, SIMPL (Scalable Iterative Maximisation of Population-coded Latents) is introduced, a straightforward yet effective enhancement to the current paradigm. This approach fits tuning curves to observed behaviour and refines these by iterating a two-step process. First the latent trajectory is *decoded* from the current tuning curves then, the

tuning curves are *refitted* based on this decoded latent trajectory. SIMPL imposes minimal constraints on the tuning curve structure, scales well to large datasets without relying on neural networks that can be expensive to train. Theoretical analysis establishes formal connections to expectation-maximisation (EM, Dempster et al. (1977)) for a flexible class of generative models. By exploiting behaviour as an initialisation, SIMPL converges fast and helps mitigate local minima and identifiability (Hyvärinen et al. 1999; Locatello et al. 2019) issues. This allows it to reliably return refined tuning curves and latents which remain close to, but improve upon, their behavioural analogues readily admitting direct comparison. All in all, SIMPL is able to identify temporally smooth latents and complex tuning curves related to behaviour, while remaining cheap and natively supporting spiking data — a distinguishing set of features in the field of latent variable models for neural data analysis.

SIMPL is first validated on a dataset of synthetically generated 2D grid cells. Next, I apply SIMPL to rodent electrophysiological hippocampal data (Tanni et al. 2022) and show that it modifies the latent space in an incremental but significant way: optimised tuning curves are better at explaining held-out neural data and contain sharper, more numerous place fields allowing for a reinterpretation of previous experimental results. Finally, SIMPL is applied to a somatosensory dataset for a macaque performing a centre-out reaching task (Chowdhury et al. 2020). SIMPL, with a 4D latent space, provides a good account of the data with the latent variables initialised to (and remaining correlated with) the macaque's hand-position and hand-velocity. With only two hyperparameters, SIMPL can be run quickly on large neural datasets[1] without requiring a GPU. It outperforms popular alternative techniques based on neural networks (Schneider et al. 2023; Zhao et al. 2017) or Gaussian processes (Lawrence 2003; Wang et al. 2005) and is over 15× faster. This makes it a practical alternative to existing tools particularly of interest to navigational or motor-control communities where abundant data is explained

[1] One-hour recordings of 200 neurons (10^6 spikes) takes 1 minute to run on a CPU laptop.

well by measurable behaviours (position, hand dynamics). An open-source JAX-optimised (Bradbury et al. 2018) implementation of the code is provided (see Appendix E.1).

5.2 The SIMPL Algorithm: An EM-Style Approach

A high-level description of the SIMPL algorithm is given here. Comprehensive details and a theoretical analysis linking SIMPL to expectation-maximisation, are provided in Appendix E.3.

5.2.1 The Model

SIMPL models *spike trains* of the form $\mathbf{s} \coloneqq (s_{ti})_{t=1,\dots T}^{i=1,\dots N}$, where s_{ti} represents the number of spikes emitted by neuron i between time $(t-1)\cdot dt$ and $t \cdot dt$. I denote $\mathbf{s}_t \coloneqq (s_{t1}, \dots, s_{tN})$ the vector of spike counts emitted by all neurons in the t-th time bin. SIMPL posits that such spike trains $\mathbf{s}$ are modulated by a *latent, continuously-valued, low-dimensional, time-evolving* variable $\mathbf{x} \coloneqq (\mathbf{x}_t)_{t=1,\dots,T} \in \mathbb{R}^D$ through the following random process:

$$\mathbf{x}_{t+1} \mid \mathbf{x}_t \sim \quad \mathcal{N}(\mathbf{x}_t, \sigma_v^2 \mathbf{I}) \qquad \text{(Latent dynamics)} \tag{5.1}$$

$$s_{ti} \mid \mathbf{x}_t \sim \quad \text{Poisson}(f_i(\mathbf{x}_t)) \qquad \text{(Emission model)} \tag{5.2}$$

where $\sigma_v \coloneqq \mathrm{v} \cdot \mathrm{d}t$ and $\mathbf{x}_0 \sim \mathcal{N}(0, \sigma_0^2 \mathbf{I})$. This generative model enforces a tunable (through the velocity hyperparameter v) amount of temporal smoothness in the trajectories. At each time step the latent variable $\mathbf{x}_t$ determines the instantaneous firing rate of all neurons via their intensity functions f_i (hereon called *tuning curves*, collectively denoted $\mathbf{f}$), which are unknown a priori, and which SIMPL will estimate. Moreover, the common assumption is made that all neurons are *conditionally independent* given $\mathbf{x}_t$, i.e. $p(\mathbf{s}_t|\mathbf{x}_t) = \prod_{i=1}^N p(\mathbf{s}_{ti}|\mathbf{x}_t)$. Finally, it is assumed that the latent variable $\mathbf{x}$ is Markovian, a common assumption in the neuroscience literature. This model has been previously studied in the literature (Smith et al. 2003; Macke et al. 2011), albeit using

highly restrictive tuning curve models, something that SIMPL avoids.

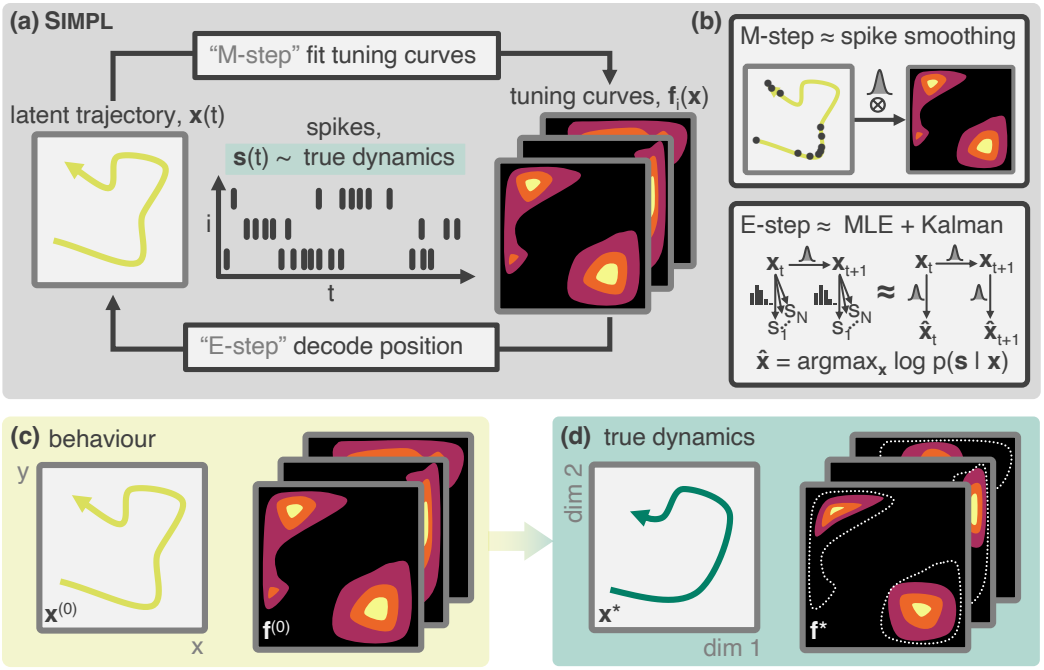

Figure 5.1: Schematic of SIMPL. **(a)** A latent variable model (LVM) for spiking data $(\mathbf{f}_i(\mathbf{x}), \mathbf{x}(t))$ is optimised by iterating a two-step procedure closely related to the expectation-maximisation: First, tuning curves are fitted to an initial estimate of the latent trajectory (an "M-step"). The latent is then *re*decoded from these tuning curves (an "E-step"). **(b)** SIMPL fits tuning curves using a kernel-smoothed estimate (top) and decodes the latent variables by Kalman-smoothing maximum likelihood estimates. **(c)** Measured behaviour is used to initialise the algorithm as it is often closely related to the true generative LVM **(d)**.

5.2.2 The SIMPL Optimisation Algorithm

Outline I now seek an estimate of the true, unknown latent trajectory $\mathbf{x}^\star$ and tuning curves $\mathbf{f}^\star$ that led to an observed spike train, $\mathbf{s}$. SIMPL does so by iterating a two-step procedure closely related to the expectation-maximisation (EM) algorithm: first, tuning curves are fitted to an initial estimate of the latent variable (the "M-step"), which are then used to decode the latent variable (the "E-step"). This procedure is then repeated using the new latent trajectory, and so on until convergence.

The M-step In the M-step (or "fitting" step) of the e-th iteration SIMPL fits tuning curves to the current latent trajectory estimate $\mathbf{x}^{(e)}$ using a smooth,

kernel-based estimate

$$f_i^{(e)}(\mathbf{x}) := \frac{\sum_{t=1}^{T} s_{ti}\, k(\mathbf{x}, \mathbf{x}_t^{(e)})}{\sum_{t=1}^{T} k(\mathbf{x}, \mathbf{x}_t^{(e)})} \approx \frac{\#\text{ spikes at x}}{\#\text{ visits to x}} \tag{5.3}$$

for some kernel k. In practice, I use a Gaussian kernel with small bandwidth σ. Such a tuning curve model is conceptually simple and free from the optimisation, misspecification or interpretability issues of most parametric models. It constitutes a notable departure from alternatives which use a neural network (Zhou et al. 2020; Schneider et al. 2023) to model tuning curves and is particularly well suited to low-dimensional latent spaces.

The E-step In the E-step SIMPL seeks to infer (or "decode") a new estimate of the latent from the spikes and current tuning curves, $\mathbf{x}^{(e+1)} = \mathbb{E}_{p(\mathbf{x}|\mathbf{s},\mathbf{f}^{(e)})}[\mathbf{x}]$. Directly performing this inference from the spikes is difficult due to the non-linearity and non-Gaussianity of the emission model in eq. (5.2). Instead, SIMPL first calculates the *maximum likelihood estimate* (MLE) of $\mathbf{x}$, denoted $\widehat{\mathbf{x}}$. Then, by making a linear-Gaussian approximation to $p(\widehat{\mathbf{x}}_t|\mathbf{x}_t) \approx \mathcal{N}(\mathbf{x}_t; \mathbf{\Sigma}_t)$, the variables $(\mathbf{x}, \widehat{\mathbf{x}})$ form a Linear Gaussian State Space Model (LGSSM) fully characterised by $\sigma_v^2\mathbf{I}$ (the transition noise covariance) and $\mathbf{\Sigma}_t$ (the observation noise covariance). This enables efficient inference via *Kalman smoothing* of the MLEs in order to approximate $\mathbf{x}^{(e+1)} = \mathbb{E}_{p(\mathbf{x}|\widehat{\mathbf{x}})}[\mathbf{x}]$ (schematic in fig. 5.1b).

$$\begin{aligned} \widehat{\mathbf{x}}^{(e+1)} &:= \arg\max_{\mathbf{x}} \log p(\mathbf{s}|\mathbf{x}, \mathbf{f}^{(e)}) \\ \mathbf{x}^{(e+1)} &:= \mathbb{E}_{p(\mathbf{x}|\widehat{\mathbf{x}}^{(e+1)})}\left[\,\mathbf{x}\,\right] \approx \text{KalmanSmooth}(\widehat{\mathbf{x}}^{(e+1)}; \sigma_v^2\mathbf{I}, \mathbf{\Sigma}_t) \end{aligned} \tag{5.4}$$

Crucially, the linear-Gaussian approximation is *not* made on the spiking emissions $p(\mathbf{s}|\mathbf{x})$, which is non-linear and non-Gaussian by design, but on $p(\widehat{\mathbf{x}}|\mathbf{x})$, a quantity which is provably asymptotically Gaussian in the many-neurons regime (theoretical argument and an explicit formula for $\mathbf{\Sigma}_t$ in Appendix E.3.1).

Behavioural initialisation Spike trains often come alongside behavioural recordings $\mathbf{x}^b$ thought to relate closely to the true latent variable $\mathbf{x}^b \approx \mathbf{x}^\star$.

SIMPL leverages this by setting the initial decoded latent trajectory, to measured behaviour $\mathbf{x}^{(0)} \leftarrow \mathbf{x}^b$. I posit that *behavioural initialisation* will place the first iterate of SIMPL within the vicinity of the true trajectory and tuning curves, accelerating convergence and favouring the true latent and tuning curves $(\mathbf{x}^\star, \mathbf{f}^\star)$ over alternative isomorphic pairs $(\phi(\mathbf{x}^\star), \mathbf{f}^\star \circ \phi^{-1})$ whose latent space is *warped* by an invertible map ϕ but which would explain the data equally well. This amounts to an inductive bias favouring tuning curves close to those calculated from behaviour. Through ablation studies I confirm these beneficial effects.

All in all, SIMPL is interpretable and closely matches common practice in neuroscience (e.g. kernel-based curve fitting, MLE-based decoding); moreover, it can be formally related to a generalised version of the EM-algorithm, for which theoretical guarantees may be obtained. I leave to Appendix E detailed theoretical arguments justifying the validity of SIMPL as well as its connection to EM.

Algorithm 1 SIMPL: An algorithm for optimizing tuning curves and latents from behaviour

$\mathbf{s} \in \mathbb{N}^{N \times T}$ ▷ Spike count matrix
$\mathbf{x}^{(0)} \in \mathbb{R}^{D \times T}$ ▷ Initial latent estimate e.g. measured position of animal
procedure SIMPL($\mathbf{s}, \mathbf{x}^{(0)}$)
    for $e \leftarrow 0$ to E **do** ▷ Loop for E iterations
        $\mathbf{f}^{(e)} \leftarrow \text{FitTuningCurves}(\mathbf{x}^{(e)}, \mathbf{s})$ ▷ The "M-step"
        $\mathbf{x}^{(e+1)} \leftarrow \text{DecodeLatent}(\mathbf{f}^{(e)}, \mathbf{s})$ ▷ The "E-step"
    end for
    return $\mathbf{x}^{(E+1)}, \mathbf{f}^{(E)}$ ▷ The optimised latent and tuning curves
end procedure

5.3 Validating SIMPL on Synthetic and Biological Data

5.3.1 Synthetic Data: 2D Grid Cells

SIMPL was first tested on a realistic navigational task by generating a large artificial dataset of spikes from a population of $N = 225$ 2D grid cells — a

type of neuron commonly found in the medial entorhinal cortex (Hafting et al. 2005) — in a 1 m square environment. All grid cells had a maximum firing rate of 10 Hz and were arranged into three discrete modules, 75 cells per module, of increasing grid scale from 0.3–0.8 m (Figure 5.2c). A latent trajectory, $\mathbf{x}^\star$, was then generated by simulating an agent moving around the environment for 1 hour under a smooth continuous random motion model. Data was sampled at a rate of 10 Hz giving a total of $T = 36,000$ time bins ($\sim$ 800,000 spikes). All data was generated using the `RatInABox` package (George et al. 2024).

The initial trajectory, $\mathbf{x}^{(0)}$, was generated by adding smooth Gaussian noise to the latent such that, on average, the true latent and initial condition differed by 20 cm (fig. 5.2a, top panel). This discrepancy models the agent's internal position uncertainty and/or a measurement error. It sufficed to obscure almost all structure from the initial tuning curves $\mathbf{f}^{(0)}(\mathbf{x})$ (fig. 5.2b, top). To assess performance I track the log-likelihood of training and test spikes (see Appendix E.4.5 for how I partition the dataset). I also calculate the error between the true and latent trajectory the epoch-to-epoch change in the tuning curves and the negative entropy (hereon called "spatial info") of the normalized tuning curves as a measure of how spatially informative they are (fig. 5.2d).

SIMPL was then run for 10 epochs (total compute time 39.8 CPU-secs on a consumer grade laptop). The true latent trajectory and receptive fields were recovered almost perfectly and the log-likelihood of both train and test spikes rapidly approached the ceiling performance with negligible overfitting. As expected, SIMPL performs better on larger datasets, fig. 5.2e, however performance remains good even with substantially smaller datasets (e.g. 50 cells for a duration of 5 minutes). A sweep was also performed across the velocity and kernel bandwidth hyperparameters (v, σ) and SIMPL was found to be surprisingly robust to changes in these hyperparameters within reasonable limits (see Appendix E.5.2).

Finally, despite having an implicit prior for temporally-smooth latent dynamics, further synthetic analysis revealed SIMPL is still able recover

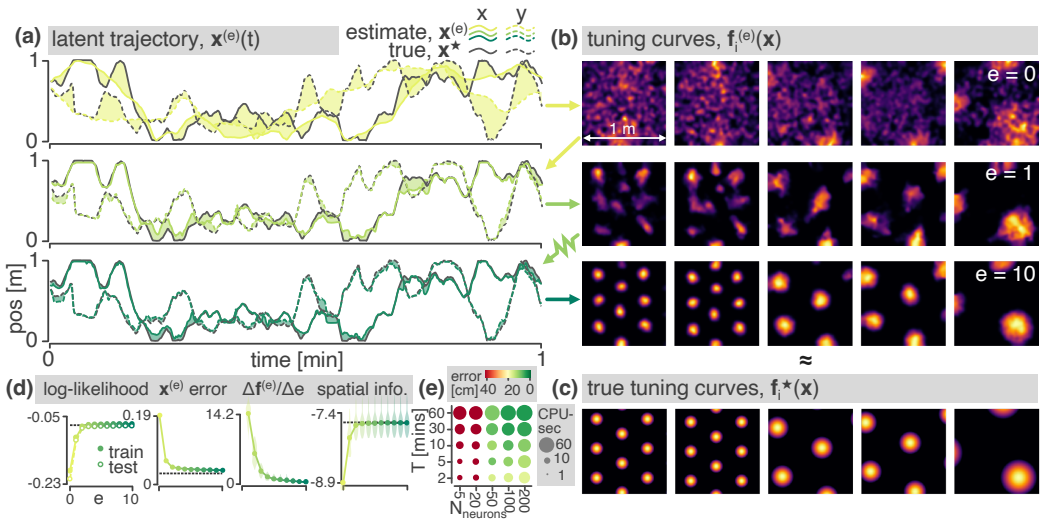

Figure 5.2: Results on a synthetic 2D grid cell dataset. **(a)** Estimated latent trajectories (epochs 0, 1 and 10). Initial conditions are generated from the true latent (black) by the addition of slow Gaussian noise. Shaded zones show the discrepancy between the true and estimated latent. **(b)** Tuning curve estimates for 5 exemplar grid cells. **(c)** Ground truth tuning curves. **(d)** Performance metrics: *Left:* log-likelihood of the training and test spikes (averaged per time step, dotted line shows ceiling performance on a model initialised with the true latent). *Middle-left:* Euclidean distance between the true and estimated latent trajectories (averaged per time step). *Middle-right:* Epoch-to-epoch change in the tuning curves showing they stabilise over iteration. *Right:* Cell spatial information. Violin plots, where shown, give distributions across all neurons. **(e)** A sweep over the number of cells and the duration of the trajectory.

discontinuous latent trajectories (for example those containing jumpy-like "replay" events, see Appendix E.5.3) or even *discrete* latents in a non-dynamical task akin to a discrete two-alternative forced choice task (2AFC, see Appendix E.5.1).

5.3.2 Hippocampal Place Cell Data

Having confirmed the efficacy of SIMPL on synthetic data, it was next tested on a real dataset of hippocampal neurons recorded from a rat as it foraged in a large environment (Tanni et al. 2022). This dataset consists of $N = 226$ neurons recorded over 2 hours, binned at 5 Hz giving $T = 36,000$ data samples and $\sim$ 700,000 spikes. Many of these cells are place cells (O'Keefe et al. 1971) which, in large environments, are known to have multiple place fields (Park et al. 2011).

I initialised with the animal's position, as measured by an LED located between its ears, and optimised for 10 epochs. The log-likelihood of test and train spikes both increased, converging after 4 epochs (fig. 5.3b) in a compute time of ∼40 CPU-secs. I then analysed the shapes and statistics of the tuning curves: After optimisation, tuning curves were visibly sharper, fig. 5.3a; previously diffuse place fields contracted (e.g. the third exemplar tuning curve) or split into multiple, smaller fields (second exemplar). Occasionally, new place fields appeared (fourth exemplar) or multiple place fields merged into a single larger field (fifth exemplar). Statistically, tuning curves had significantly more individual place fields (+19%, mean 1.14→1.41 per cell, $p = 0.0035$ Mann Whitney U tests), substantially higher maximum firing rates (+45%, median 4.2→6.1 Hz, $p = 9.8 \times 10^{-7}$) and were more spatially informative ($p = 0.038$). Individual place fields became smaller (-25%, median 0.59→0.44 m^2) and rounder (+8%, median 0.63→0.68, $p = 0.0037$).

To ensure these observed changes weren't merely an artefact of the optimisation procedure I generated a control dataset by resampling spikes from the behaviour-fitted tuning curves, $\mathbf{s}_{\text{con}} \sim p(\cdot|\mathbf{x}^{(0)}, \mathbf{f}^{(0)})$. Control spikes thus had very similar temporal statistics and identical tuning curves to those in the hippocampal dataset but, critically, were generated from a known ground truth model exactly equal to their initialization. Thus, any changes in the control tuning curves post-SIMPL must be artefactual. Indeed, no significant changes were observed besides a slight *increase* in field area (fig. 5.3bc, grey) providing strong evidence the significant changes observed in the real data (e.g. the *decrease* in field area) were genuine, reflecting the true nature of hippocampal tuning curves.

The optimised latent trajectory $\mathbf{x}^{(10)}$ remained highly correlated with behaviour ($R^2 = 0.86$, fig. 5.3d) occasionally diverging for short periods as it "jumped" to and from a new location, as if the animal was mentally teleporting itself (an example is visualized in fig. 5.3e). I calculated the difference between the optimised latent and the behaviour at each time point, $\Delta_t = \|\mathbf{x}_t^{(0)} - \mathbf{x}_t^{(10)}\|_2$,

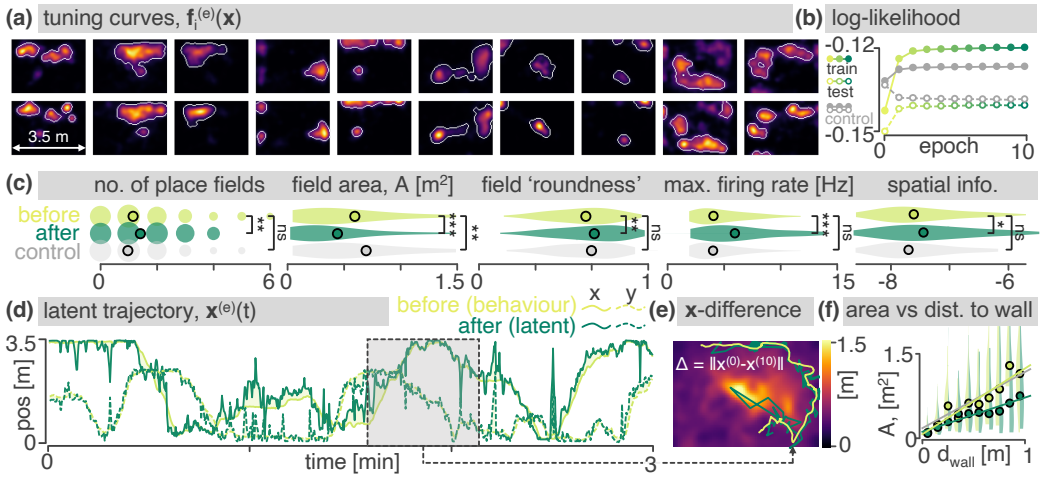

Figure 5.3: Results on a hippocampal place cell dataset collected by Tanni et al. (2022). **(a)** Exemplar tuning curves before and after optimization. Automatically identified place field boundaries shown in white. **(b)** Log-likelihood of test and train spikes. Control model shown in grey. **(c)** Statistics analysis of place fields. Violin plots show the distributions over all place fields / cells. **(d)** The final latent trajectory estimated from SIMPL (green) overlaid on top of the measured position of the animal (used as initial conditions, yellow). **(e)** Behavioural discrepancy map: the average discrepancy between the latent and behaviour as a function of the optimised latent $\mathbf{x}^{(10)}$. Overlaid is a snippet of the behavioural vs optimised true latent trajectory. **(f)** Place field area as a function of the distance to the nearest wall.

and visualized this as a heat map overlaid onto the latent space (fig. 5.3e). I found that the latent discrepancy was minimal near the edges of the environment and peaked near the centre, perhaps because sensory input is scarce in the centre of the environment due to fewer visual and tactile cues.

Tanni et al. (2022) observed that the size of a place field increases with its distance to a wall. This observation—that the latent discrepancy is highest in the centre of the environment—suggests one possible hypothesis: behavioural place fields merely *appear* larger in the centre of the environment because they are blurred by the correspondingly larger latent discrepancy. If true, this trend should weaken after optimisation, once the "true" latent has been found.

To test this I plotted field size against distance-to-wall (fig. 5.3f); optimised fields, like behavioural fields, were small very near to the walls and grew with distance (replicating the result of Tanni et al. (2022)), but this correspondence stopped after $\sim$ 0.5 m beyond which the optimised place fields size grew

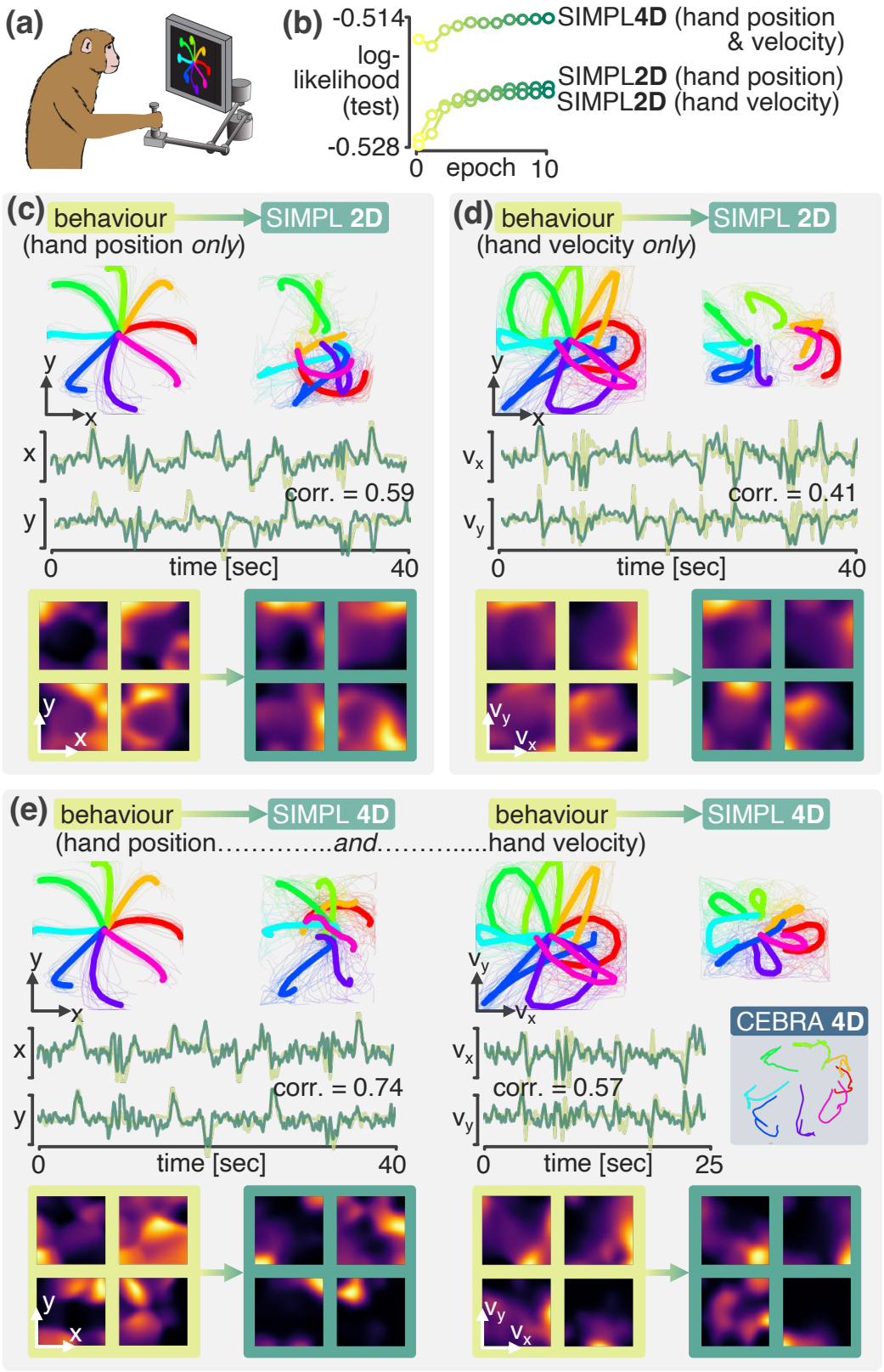

Figure 5.4: Somatosensory Cortex Results *(Continued on next page...)*

Figure 5.4: *(...continued from previous page)* SIMPL applied to somatosensory cortex data. **(a)** A macaque performs centre-out reaches; $N = 65$ somatosensory neurons are recorded. **(b)** Log-likelihood curves for the three SIMPL models in panels c–e. **(c)** SIMPL trained with a 2D latent initialised from hand position. Top-left: raw behaviour, averaged across trials aligned to movement onset; top-right: after SIMPL. Middle: 40 s of behaviour (yellow) and latent (green). Bottom: exemplar tuning curves before and after SIMPL. **(d)** As in c, but initialised with hand velocity. **(e)** As in c, but with a 4D latent initialised to hand-position (dims 1 and 2) and velocity (dims 3 and 4). Inset: 2D visualisation of a 4D latent embedding from CEBRA trained on hand position, adapted from Schneider et al. (2023).

more weakly with distance-to-wall. This supports my hypothesis, suggesting a substantial fraction of the correlation between size and distance isn't a fundamental feature of the neural tuning curves but an artefactual distortion in the tuning curves, something which can be corrected for using SIMPL.

5.3.3 Somatosensory Cortex Data During a Hand-Reaching Task

To test SIMPL beyond navigational/hippocampal datasets, it was run on a macaque somatosensory cortex dataset Chowdhury et al. (2020). During this recording a macaque made a series of reaches to a target in one of 8 directions, fig. 5.4. On half of the trials the reach was "active" whereby the macaque moved the manipulandum towards the target by itself. On the other half, the reach was "passive", whereby the macaque's hand was bumped in the direction of one of the targets by a force applied to the manipulandum, forcing the macaque to correct and return the cursor to the centre. I binned the data ($N = 65$ neurons, 37 mins, $\sim 10^6$ spikes) at 20 Hz and ran SIMPL models on its entirety (i.e. active and passive reaches, as well as the inter-trial intervals) for 10 epochs.

First SIMPL was run with a 2D latent initialised to the macaque's measured x- and y-hand position (fig. 5.4c). Afterwards, the latent trajectory—here averaged across trials with the same direction, aligned to movement onset—had diverged from, but remained correlated with, initial hand-position (correlation = 0.59). Despite an improvement in likelihood over the behavioural initialisation,

latent trajectories for distinct directions substantially overlapped with one another, indicating an insufficient dimensionality to capture the full complexity of the data. A similar result was obtained when initialising to hand-velocity (fig. 5.4d).

I then trained SIMPL with a 4D latent space. Two of the dimensions were initialised with hand position and the other two with hand velocity. This model performed better than either 2D model, converging to a higher likelihood. The latent dimensions initialised to hand-position remained highly correlated with hand-position (corr. = 0.74) after optimisation as did the velocity dimensions (corr. = 0.57). The latent trajectory was also more structured, with distinct and less overlapping motifs for each trial type. I visualised two-dimensional slices of the four-dimensional tuning curves for each neuron and found that they had well-defined receptive fields, similar to place fields in the hippocampus, which were visibly sharper after optimisation. These results suggest that the somatosensory cortex neurons encode a complex and high-dimensional latent, closely correlated to hand position and velocity, which can be partially recovered by SIMPL.

5.3.4 The Critical Role of Behavioral Initialization

Latent variable models trained with EM can experience two issues that usually complicate the scientific interpretability of their results. The first concerns the *quality* of the solution; does the algorithm converge on a good model of the data which predicts the spikes well? The second issue concerns *identifiability*; even if the recovered latent trajectory and tuning curves $(\mathbf{f}^{(e)}, \mathbf{x}^{(e)})$ are of high quality, they may differ from the true ones $(\mathbf{f}^{\star}, \mathbf{x}^{\star})$ by some invertible "warp" ϕ in a way that does not affect the overall goodness-of-fit of the model. These warps could include innocuous rotations and symmetries or, more concerningly if the exact structure of the tuning curve is a quantity of interest, stretches or fragmentations. Here it is shown that behavioural initialisation drastically minimises the severity of both of these issues for SIMPL.

To do so, I first assess the absolute goodness-of-fit of SIMPL by computing,

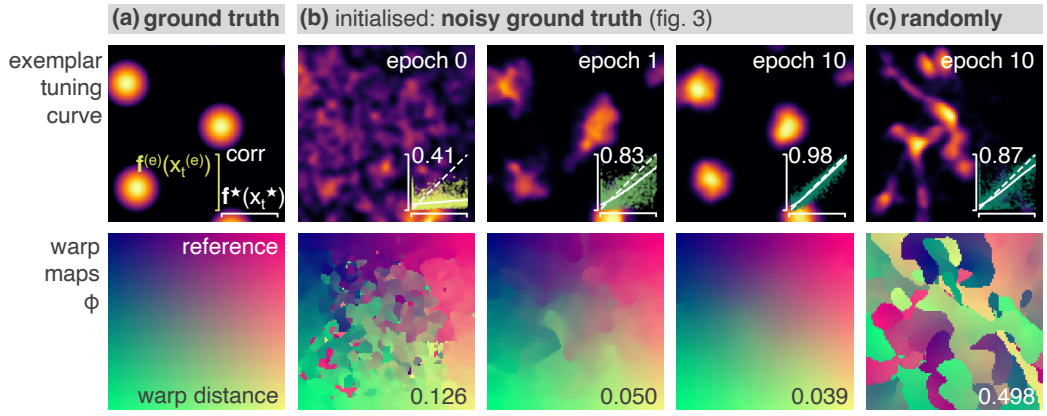

Figure 5.5: Latent manifold analysis: **(Top)** Exemplar tuning curve in **(a)** the ground truth latent space, **(b)** the latent space discovered by behaviourally-initialised-SIMPL after 0, 1 and 10 epochs and **(c)** the latent space discovered by SIMPL initialised with a random latent trajectory. Inset scatter plots show the true and predicted firing rates of all neurons across all times as well as their correlation values ("accurate" models have higher correlations). **(Bottom)** The warp mappings from each latent space to the "closest" location in ground truth as measured by the distance between the tuning curves population vectors.

for all neurons, the correlation between the estimated instantaneous firing rates $f_i^{(e)}(\mathbf{x}_t^{(e)})$ (a quantity invariant to warping) and the true firing rates $f_i^\star(\mathbf{x}_t^\star)$. SIMPL converges to a highly accurate model (r=0.98) under behavioural initialization, but to a less accurate, though still quite accurate, model ($r = 0.87$) when initialised with a random trajectory uncorrelated to the true latent. Next, I estimate, quantify and visualize the warp map ϕ between SIMPL's estimates $(\mathbf{f}^{(e)}, \mathbf{x}^{(e)})$ and the ground truth $(\mathbf{f}^\star, \mathbf{x}^\star)$. I obtain this by finding, for every location in the warped space, the position in the true latent space where the tuning curves are most similar ($\phi(\mathbf{x}) = \arg\min_{\mathbf{y}} \|\mathbf{f}^\star(\mathbf{y}) - \mathbf{f}^{(e)}(\mathbf{x})\|_2$). I then quantify the "warpness" of this mapping as the average distance between $\mathbf{x}$ and $\phi(\mathbf{x})$ across the environment, normalized by its characteristic length scale (1 m). This warp-distance should be 0 for totally un-warped models and $\mathcal{O}(1)$ for heavy warps. In addition to perfectly fitting the data, the solution found by SIMPL under behavioural initialization is minimally warped (warp dist = 0.050). In contrast, the good (but imperfect) solution found by SIMPL under random initialization is very heavily warped (warp dist. = 0.498) in

a fragmented manner. These results (fig. 5.5) strongly motivate the use of behavioural initializations in latent variable models as an effective means to encourage convergence towards latent spaces which are both accurate and un-warped with respect to the ground truth.

5.3.5 Performance Comparison with State-of-the-Art Methods

I compared SIMPL to four popular methods for latent variable extraction: pi-VAE (Zhou et al. 2020), CEBRA (Schneider et al. 2023) (that use neural network function approximators), GPLVM (Lawrence 2003) and GPDM (Wang et al. 2005) (that use Gaussian processes). Crucially, and like SIMPL, none of these methods make restrictive linear assumptions about the structure of the tuning curves.

To match SIMPL, I initialise the latent variable estimates of GPLVM and GPDM to behaviour (pi-VAE and CEBRA handle behaviour natively by using it to condition a prior over the latent or as a contrastive label). All models were trained for their default number of iterations/epochs. After training I aligned the discovered latents to behaviour and visualised them on top of the ground truth (fig. 5.6c). All models successfully uncovered a latent trajectory closer to the ground truth than behaviour (fig. 5.6b). SIMPL performed better than the other models, achieving a final error of 4.2 cm, half that of pi-VAE (8.4 cm).

I posit that pi-VAE, CEBRA and GPLVM may suffer from the lack of an explicit dynamical systems component in their generative models while GPDM may suffer from the data-subsampling that was required to cap the training time to less than two-hours. SIMPL converged in 40 seconds, over 15 times quicker than the next fastest (pi-VAE, 10.4 minutes, fig. 5.6a). Except for GPDM, which required a GPU, all techniques were run and timed on a CPU. Only SIMPL was able to recover sharp and accurate grid fields close to the ground truth.

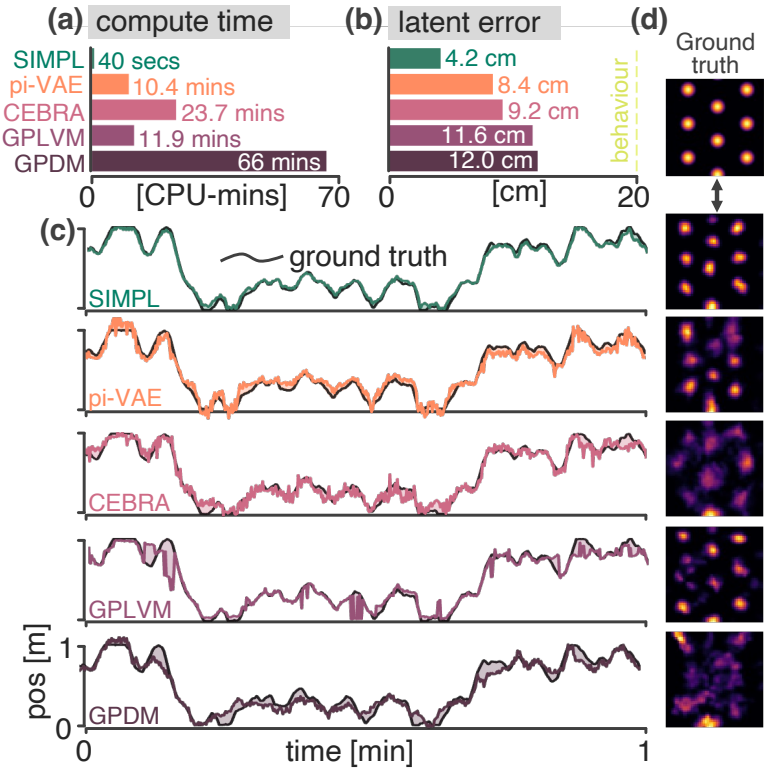

Figure 5.6: Comparison to pi-VAE, CEBRA, GPLVM and GPDM on the synthetic grid cell dataset. **(a)** Compute time. **(b)** Final error in the latent. **(c)** Alignment of the discovered latent to the ground truth. **(d)** Exemplar tuning curves constructed using kernel-based estimation on the latent (i.e. an "M-step").

5.4 A Survey of Latent Variable Models for Neural Data

Probabilistic inference in neural data modulated by latent variables has been a major topic of study for decades — see, e.g. Tipping et al. (1999), Yu et al. (2006), Yu et al. (2008b), Yu et al. (2008a), Macke et al. (2011), Mangion et al. (2011), Park et al. (2015), Gao et al. (2016), Hernandez et al. (2018), Dong et al. (2020), Zhou et al. (2020), Gondur et al. (2023), and Bjerke et al. (2023) — however not all methods were designed for the kind of data considered in this chapter. Many methods model complex latent space dynamics but combine these with simplistic tuning curves that restrict firing rates to (exponential-)linear functions of the latent (Smith et al. 2003; Yu et al. 2008a; Macke et al. 2011; Duncker et al. 2019; Linderman et al. 2016; Pandarinath et al. 2018;

Zoltowski et al. 2020; Sani et al. 2021; Hurwitz et al. 2021; Kim et al. 2021; Gondur et al. 2023) so cannot interpretably account for the representations (place cells, grid cells) considered here. Other methods do not/cannot use behaviour to aid latent discovery (Gao et al. 2016; Nam 2015; Hernandez et al. 2018; Gondur et al. 2023; Bjerke et al. 2023) instead taking a fully "unsupervised" approach (meaning they can be applied to spike data without an obvious behavioural correlate) at the expense of complexity and identifiability.

Algorithms that both don't restrict to simplistic linear tuning curves and exploit behaviour form a small set of relevant alternatives to SIMPL. Behaviour-informed latent discovery tools have become popular in recent years due to the explosion of large neural datasets taken from behaving animals and the observation that behaviour can explain substantial variance in the neural dynamics.

Gaussian process latent variable models (GPLVMs), Lawrence (2003) and Wang et al. (2005) form a family of methods that learn smooth, non-linear tuning curves by placing GP priors on them and performing approximate marginal log-likelihood optimisation on the latent variable. Popular implementations leave the initial condition of this optimisation user-defined and therefore compatible with the behaviour-informed initialisation used here. However, most such models were introduced outside of the neuroscience literature thus use Gaussian (instead of Poisson) emission models (Lawrence 2003; Wang et al. 2005; Jensen et al. 2020), or do not make smoothness assumptions on the latent trajectory (Jensen et al. 2020; Lawrence 2003). P-GPLVM, which employs Poisson emissions and a GP prior on the latent trajectory, is an exception, but its cubic scaling with time points makes it impractical for hour-long datasets. In contrast, available GPLVM implementations (Bingham et al. 2018) use inducing point approximations to achieve linear time complexity.

CEBRA (Schneider et al. 2023) learns a deterministic neural network mapping from spikes to latents using behaviour- or time-guided contrastive learning. Unlike most methods, CEBRA does not natively learn a generative

model nor tuning curves, which are of primary interest in this setting. CEBRA also treats each data point independently instead of modelling whole-trajectories preventing it from taking advantage of the temporal smoothness inherent in many underlying latent codes.

pi-VAE (Zhou et al. 2020) uses a variational autoencoder (Kingma et al. 2014) to infer the latent trajectories and learn tuning curves using neural network function approximators. pi-VAE places a learnable prior, conditioned on behaviour, to the latent variable in order to obtain a model with provable identifiability properties. However, pi-VAE suffers from the same limitation as CEBRA in that it treats each data point as an i.i.d observation instead of a part of a whole trajectory.

The properties of large scale neural datasets suggest five desiderata for the algorithms used to analyse them. These are (1) the absence of restrictive tuning curve assumptions, (2) modelling smooth latent dynamics, (3) the presence of a spiking component (e.g. Poisson emissions), (4) the ability to exploit behaviour (including as an initial condition) and (5) scalability to large datasets. None of the methods described in the literature review satisfy all five desiderata. In Appendix E.6, a table is provided comparing all methods discussed in this section and more with respect to these desiderata.

5.5 Discussion: A Practical Tool for Uncovering the Brain's Latent Code

SIMPL was introduced, a tool for optimizing tuning curves and latent trajectories using a technique that refines estimates obtained from behaviour. It hinges on two well-established sub-routines — tuning curve fitting and decoding — that are widely used by both experimentalists and theorists for analysing neural data. By presenting SIMPL as an iterative application of these techniques, I aim to make latent variable modelling more accessible to the neuroscience community.

SIMPL could be seen as an instance of a broader class of latent optimization

algorithms. In principle *any* curve fitting procedure and *any* decoder (which uses those tuning curves) could be coupled into a candidate algorithm for optimizing latents from neural data. My specific design choices, while attractive due to their conceptual and computational simplicity, will come with limitations. For example, I predict SIMPL's kernel-based estimator won't scale well to very high dimensional latent spaces (Györfi et al. 2006) where parametric models, e.g. a neural networks, are known to perform better (Bach 2017), potentially at the cost of compute time.

My synthetic analysis focused on settings where behaviour and the true latent differed only in an unbiased manner. It would be interesting to determine if SIMPL's performance extends to more complex perturbations. Fast, non-local and asymmetric perturbations are common in the brain; for instance "replay" (Carr et al. 2011) where the latent jumps to another location in the environment. Likewise, during theta sequences (Maurer et al. 2006), the encoded latent moves away from the agent. This forward-biased discrepancy could theoretically induce a backward-biased skew in behavioural place fields, even if the true tuning curves remain unskewed. If this is the case, proper latent dynamical analysis—via tools like SIMPL—could help reinterpret the predictive nature of place field tuning curves (Stachenfeld et al. 2017; Fang et al. 2023; Bono et al. 2023; George et al. 2023a), similar to how it reduced the asymmetry in place field sizes further from walls (fig. 5.3f).

Chapter 6

General Discussion

This thesis has explored how neural systems learn and use internal representations to support flexible behaviour, with a primary focus on the mammalian spatial memory system. By developing computational tools and biologically plausible models, this work studies the learning of, and interplay between, structure and dynamics that gives rise to cognitive maps with sophisticated functions. The research advances both theoretical and practical understanding within neuroscience and contributes new tools and models to the growing field of NeuroAI (Zador et al. 2023).

6.1 Summary Discussion of the Major Themes

A central theme of this thesis has been **the development and analysis of computational models that are explicitly designed to be biologically plausible**, serving as a bridge between high-level cognitive theories and the mechanistic details of neural circuits. The work presented in Chapters 2 and 3 confronts this challenge directly by proposing a plausible mechanism for how the hippocampus could learn a predictive map, a concept from reinforcement learning. Instead of relying on error-driven algorithms, these models demonstrate that spike-timing dependent plasticity (STDP), when combined with the temporal compression afforded by theta phase precession, is sufficient to learn a close approximation of the successor representation. The work in Chapter 4 introduces a generative model of the hippocampal-

entorhinal loop that performs path integration and mental simulation using only local, Hebbian-style learning rules gated by theta oscillations, avoiding unrealistic requirements like backpropagation-through-time and other non-local learning mechanisms. Collectively, these models provide an existence proof that sophisticated and theoretically powerful computations are achievable within the known anatomical and physiological constraints of the hippocampal formation.

Complementing this theoretical work, a second major theme was **the development and validation of practical, open-source tools to enable better and more reproducible research**. This is most directly addressed in Chapter 1, which introduces the *RatInABox* toolkit. This software provides a standardised, efficient, and realistic environment for simulating rodent locomotion and neural activity, tackling the issues of duplicated effort and a lack of comparability between studies. Furthering this theme, Chapter 5 presents SIMPL, a novel and computationally efficient algorithm for latent variable discovery. By leveraging behaviour as an initial condition, SIMPL offers a practical solution to the long-standing challenge of characterising neural tuning curves, revealing that the brain's internal representations may be significantly more precise than what is observable from behaviour alone. These tools are not merely by-products of the research but are themselves primary contributions designed to strengthen the infrastructure of computational neuroscience as applied to the spatial memory system.

6.2 Open Questions and Future Research Themes

While by no means exhaustive, several key themes and open questions for future research emerge from the work presented in this thesis:

1. **The Interplay Between Structural Learning and Dynamic Computation:** The thesis explores how predictive maps can be learned through long-term structural changes in synaptic weights (Chapter 3) and how neural dynamics can perform computations like path integration

(Chapter 4). A significant future direction is to understand how these two modes of computation—modifying network structure versus modulating network dynamics—are balanced and integrated. This raises the question: Under what circumstances does the brain, or another neural system, favour altering long-term synaptic connections versus relying on short-term, dynamic computations to support flexible behaviour, a question that mirrors the distinction between "in-weights" and "in-context" learning in modern AI (Brown et al. 2020).

2. **Unifying Different Forms of Experience Compression for Learning:** The work highlights how theta sequences and phase precession can compress behavioural timescales to enable learning with STDP (Chapters 2 and 3). The hippocampus also uses sharp-wave ripple (SWR) replay to compress past experiences, a process thought to be critical for memory consolidation (Wilson et al. 1994). Other fast oscillatory patterns, such as gamma oscillations, are also implicated in replay and learning (Li et al. 2021). A key open question is how these different forms of compressed replay are coordinated. Do theta sweeps and SWR replay serve distinct learning functions? For instance, theta-based learning might support online, incremental updates to a cognitive map, while SWR replay could be reserved for prioritised consolidation of particularly important experiences, such as trajectories leading to reward (Mattar et al. 2018) or exposure to novel environments (Cheng et al. 2008).

3. **Extending Latent Variable Discovery to Abstract and Non-Spatial Domains:** The SIMPL algorithm (Chapter 5) demonstrates that repeated decoding and refitting, using behaviour as an initial estimate, can successfully refine the underlying latent neural representations for spatial tasks. The hippocampus, however, is thought to encode not just physical space but also abstract and non-spatial "cognitive maps" (Constantinescu et al. 2016; Garvert et al. 2017; Whittington et al. 2020). A compelling future direction is to apply or adapt tools such as SIMPL to investigate

these abstract cognitive spaces. By initialising the model with non-spatial behavioural variables (e.g., progress toward a goal, social hierarchies, or task parameters), researchers could explore whether the underlying neural representations conform to similar principles of organisation (e.g., place-like or grid-like coding) as the spatial domain.

4. **Scaling Biologically Plausible Learning Rules for Complex Generative Models:** The generative model in Chapter 4 shows how local, Hebbian-style learning rules, gated by oscillations, can give rise to sophisticated functions like path integration through the emergence of a ring attractor structure. While this provides a crucial proof of principle, a major challenge still remains in scaling these biologically plausible mechanisms to handle the complexity and high dimensionality of real-world sensory inputs, a domain where deep learning models trained with backpropagation currently excel (Goodfellow et al. 2014; Vaswani et al. 2017). Future research could explore how architectures inspired by the hippocampal formation might be expanded to create more powerful and generalisable generative models for AI, potentially offering advantages in efficiency (Lillicrap et al. 2020) or continual learning (Kirkpatrick et al. 2017; Parisi et al. 2018).

6.3 Open science: Towards a more equitable and transparent global research culture

The open-source toolkits developed in this thesis, *RatInABox* (Chapter 1) and SIMPL (Chapter 5), are rooted in a firm commitment to open and reproducible science. As a practical extension of this philosophy, a significant part of my doctoral work has been dedicated to co-founding the *TReND-CaMinA* summer school (Soldado-Magraner et al. 2023; Cashin-Garbutt et al. 2023). This non-profit initiative, now in its fourth year, provides training in computational neuroscience and machine learning for young African researchers, addressing the issue that Africa's scientific output remains low (Nabyonga-Orem et al. 2024;

Schneegans et al. 2021) despite the continent's large and youthful population (Ashford et al. 2007). Entirely free for students and funded by donations from the global research community, the school embodies open-science principles by training students on open-source tools and large-scale public datasets (Vries et al. 2023), and making all teaching materials open-source.

By lowering barriers to research and education, the tools in this thesis and community-driven initiatives like *TReND-CaMinA* are complementary efforts toward fostering a more equitable and transparent global research culture.

6.4 Conclusion

The research presented in this thesis has been guided by a central goal: to bridge the gap between powerful, normative theories of cognition and the mechanistic realities of neural circuits. By demonstrating how fundamental biological processes like spike-timing dependent plasticity and neural oscillations can implement sophisticated computations, this work takes a tangible step towards a more unified understanding of the hippocampus. It is my hope that the models and tools developed here will not only advance our knowledge but also inspire future research that continues to ground abstract ideas in concrete, testable neural mechanisms, pushing our field closer to a more comprehensive understanding of the brain and its role in all aspects of cognition.

Bibliography

Afshar, Afsheen, Gopal Santhanam, Byron M. Yu, Stephen I. Ryu, Maneesh Sahani, and Krishna V. Shenoy (2011). "Single-trial neural correlates of arm movement preparation". In: *Neuron*. DOI: 10.1016/j.neuron.2011.05.047.

Alvernhe, Alice, Etienne Save, and Bruno Poucet (2011). "Local remapping of place cell firing in the Tolman detour task". In: *European Journal of Neuroscience* 33.9, pp. 1696–1705. ISSN: 1460-9568. DOI: 10.1111/j.1460-9568.2011.07653.x.

Ascoli, Giorgio A., Duncan E. Donohue, and Maryam Halavi (Aug. 2007). "NeuroMorpho.Org: A Central Resource for Neuronal Morphologies". In: *The Journal of Neuroscience* 27.35, pp. 9247–9251. ISSN: 1529-2401. DOI: 10.1523/jneurosci.2055-07.2007.

Ashford, Lori S., R. A. Garcia, B. S. Soares Filho, Y. Cai, R. Lakshminarayanan, J. F. May, E. Bos, R. Hasan, E. Suzuki, and T. R. Aryal (2007). "Africas Youthful Population: Risk or Opportunity?" In: *Journal of Biosocial Science* 39.5, pp. 693–706. URL: https://www.sarpn.org/documents/d0002763/PRB_Africa_youth_Jun2007.pdf.

Bach, Francis (2017). "Breaking the curse of dimensionality with convex neural networks". In: *Journal of Machine Learning Research* 19, pp. 1–53. DOI: 10.48550/arXiv.1412.8690.

Banino, Andrea, Caswell Barry, Benigno Uria, Charles Blundell, Timothy P. Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J. Chadwick, Thomas Degris, Joseph Modayil, Greg Wayne, Hubert Soyer, Fabio Viola, Brian Zhang, Ross Goroshin, Neil Rabinowitz, Razvan Pascanu, Charlie Beattie, Stig Petersen, Amir Sadik, Stephen Gaffney, Helen King, Koray Kavukcuoglu, Demis Hassabis, Raia Hadsell, and Dharshan Kumaran (May 2018). "Vector-based navigation using grid-like

representations in artificial agents". In: *Nature* 557.7705, pp. 429–433. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-018-0102-6.

Barak, Omri (Oct. 2017). "Recurrent neural networks as versatile tools of neuroscience research". In: *Current Opinion in Neurobiology* 46, pp. 1–6. ISSN: 0959-4388. DOI: 10.1016/j.conb.2017.06.003.

Barnes, Nick (Oct. 2010). "Publish your computer code: it is good enough". In: *Nature* 467.7317, pp. 753–753. ISSN: 1476-4687. DOI: 10.1038/467753a.

Barry, Caswell, Robin Hayman, Neil Burgess, and Kathryn J. Jeffery (May 2007). "Experience-dependent rescaling of entorhinal grids". In: *Nature Neuroscience* 10.6, pp. 682–684. DOI: 10.1038/nn1905.

Barry, Caswell, Colin Lever, Robin Hayman, Tom Hartley, Stephen Burton, John O'Keefe, Kathryn J. Jeffery, and Neil Burgess (2006). "The boundary vector cell model of place cell firing and spatial memory". In: *Reviews in the Neurosciences* 17.1-2, pp. 71–98. ISSN: 0334-1763. DOI: 10.1515/revneuro.2006.17.1-2.71.

Benna, Marcus K. and Stefano Fusi (2021). "Place cells may simply be memory cells: Memory compression leads to spatial tuning and history dependence". In: *Proceedings of the National Academy of Sciences* 118.51, e2018422118. DOI: 10.1073/pnas.2018422118.

Bi, Guo-qiang and Mu-ming Poo (Dec. 1998). "Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type". In: *The Journal of Neuroscience* 18.24, pp. 10464–10472. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.18-24-10464.1998.

Billingsley, Patrick (1961). "Statistical methods in Markov chains". In: *The annals of mathematical statistics*. DOI: 10.1214/aoms/1177705136.

Bingham, Eli, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman (2018). "Pyro: Deep universal probabilistic programming". In: *Journal of Machine Learning Research*. DOI: 10.48550/arXiv.1810.09538.

Bittner, Katie C., Christine Grienberger, Sachin P. Vaidya, Aaron D. Milstein, John J. Macklin, Junghyup Suh, Susumu Tonegawa, and Jeffrey C. Magee (July 2015).

"Conjunctive input processing drives feature selectivity in hippocampal CA1 neurons". In: *Nature Neuroscience* 18.8, pp. 1133–1142. DOI: 10.1038/nn.4062.

Bittner, Katie C., Aaron D. Milstein, Christine Grienberger, Sandro Romani, and Jeffrey C. Magee (Sept. 2017). "Behavioral time scale synaptic plasticity underlies CA1 place fields". In: *Science* 357.6355, pp. 1033–1036. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aan3846.

Bjerke, Martin, Lukas Schott, Kristopher T. Jensen, Claudia Battistin, David A. Klindt, and Benjamin Adric Dunn (2023). "Understanding neural coding on latent manifolds by sharing features and dividing ensembles". In: *The Eleventh International Conference on Learning Representations*. DOI: 10.48550/arXiv.2210.03155.

Bliss, T. V. P. and T. Lømo (July 1973). "Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path". In: *The Journal of Physiology* 232.2, pp. 331–356. ISSN: 1469-7793. DOI: 10.1113/jphysiol.1973.sp010273.

Blum, Kenneth I. and L. F. Abbott (Jan. 1996). "A Model of Spatial Map Formation in the Hippocampus of the Rat". In: *Neural Computation* 8.1, pp. 85–93. ISSN: 0899-7667. DOI: 10.1162/neco.1996.8.1.85.

Bogacz, Rafal, Eric Brown, Jeff Moehlis, Philip Holmes, and Jonathan D. Cohen (2006). "The physics of optimal decision making: A formal analysis of models of performance in two-alternative forced-choice tasks". In: *Psychological review*. DOI: 10.1037/0033-295x.113.4.700.

Bolding, Kevin A., Janina Ferbinteanu, Steven E. Fox, and Robert U. Muller (July 2019). "Place cell firing cannot support navigation without intact septal circuits". In: *Hippocampus* 30.3, pp. 175–191. DOI: 10.1002/hipo.23136.

Bono, Jacopo, Sara Zannone, Victor Pedrosa, and Claudia Clopath (Aug. 2021). "Learning predictive cognitive maps with spiking neurons during behaviour and replays". In: DOI: 10.1101/2021.08.16.456545.

Bono, Jacopo, Sara Zannone, Victor Pedrosa, and Claudia Clopath (2023). "Learning predictive cognitive maps with spiking neurons during behavior and replays". In: *eLife*, e80671. DOI: 10.7554/elife.80671.

Bostock, Elizabeth, Robert U. Muller, and John L. Kubie (Apr. 1991). "Experience-dependent modifications of hippocampal place cell firing". In: *Hippocampus* 1.2, pp. 193–205. ISSN: 1098-1063. DOI: 10.1002/hipo.450010207.

Bradbury, James, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang (2018). *JAX: Composable transformations of Python+NumPy programs*. version: 0.3.13. URL: https://github.com/jax-ml/jax.

Bradley, Ralph A. and John J. Gart (1962). "The asymptotic properties of ML estimators when sampling from associated populations". In: *Biometrika*. DOI: 10.2307/2333482.

Brankack, Jurij, Mark Stewart, and Steven E. Fox (July 1993). "Current source density analysis of the hippocampal theta rhythm: Associated sustained potentials and candidate synaptic generators". In: *Brain Research* 615.2, pp. 310–327. DOI: 10.1016/0006-8993(93)90043-m.

Brea, Johanni, Alexisz Tamás Gaál, Robert Urbanczik, and Walter Senn (June 2016). "Prospective Coding by Spiking Neurons". In: *PLOS Computational Biology* 12.6, e1005003. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1005003.

Bredenberg, Colin, Eero P. Simoncelli, Benjamin S. H. Lyo, and Cristina Savin (2021). "Impression learning: Online representation learning with synaptic plasticity". In: p. 13. URL: https://proceedings.neurips.cc/paper/2021/hash/615299acbbac3e21302bbc435091ad9f-Abstract.html.

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei (2020). "Language Models are Few-Shot Learners". In: DOI: 10.48550/ARXIV.2005.14165.

Bubeck, Sébastien, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg,

Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang (2023). *Sparks of artificial general intelligence: Early experiments with GPT-4*. DOI: 10.48550/arXiv.2303.12712.

Buckner, Randy L. and Daniel C. Carroll (Feb. 2007). "Self-projection and the brain". In: *Trends in Cognitive Sciences* 11.2, pp. 49–57. ISSN: 1364-6613. DOI: 10.1016/j.tics.2006.11.004.

Buhry, Laure, Amir H. Azizi, and Sen Cheng (2011). "Reactivation, Replay, and Preplay: How It Might All Fit Together". In: *Neural Plasticity* 2011, pp. 1–11. ISSN: 1687-5443. DOI: 10.1155/2011/203462.

Buonomano, Dean V. and Wolfgang Maass (Jan. 2009). "State-dependent computations: spatiotemporal processing in cortical networks". In: *Nature Reviews Neuroscience* 10.2, pp. 113–125. ISSN: 1471-0048. DOI: 10.1038/nrn2558.

Burak, Yoram and Ila R. Fiete (Feb. 2009). "Accurate Path Integration in Continuous Attractor Network Models of Grid Cells". In: *PLOS Computational Biology* 5.2, e1000291. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1000291.

Burgess, Neil, Caswell Barry, and John O'Keefe (2007). "An oscillatory interference model of grid cell firing". In: *Hippocampus* 17.9, pp. 801–812. ISSN: 1098-1063. DOI: 10.1002/hipo.20327.

Burgess, Neil, James G. Donnett, Kathryn J. Jeffery, and John O'Keefe (1997). "Robotic and neuronal simulation of the hippocampus and rat navigation". In: *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 352.1360, pp. 1535–1543. DOI: 10.1098/rstb.1997.0140.

Bush, Daniel, Caswell Barry, and Neil Burgess (2014). "What do grid cells contribute to place cell firing?" In: *Trends in Neurosciences* 37.3, pp. 136–145. ISSN: 0166-2236. DOI: 10.1016/j.tins.2013.12.003.

Bush, Daniel, Caswell Barry, Daniel Manson, and Neil Burgess (2015). "Using Grid Cells for Navigation". In: *Neuron* 87, pp. 507–520. DOI: 10.1016/j.neuron.2015.07.006.

Bush, Daniel, H. Freyja Ólafsdóttir, Caswell Barry, and Neil Burgess (2022). "Ripple band phase precession of place cell firing during replay". In: *Current Biology* 32.1, pp. 64–73. DOI: 10.1016/j.cub.2021.10.033.

Bush, Daniel, Andrew Philippides, Phil Husbands, and Michael O'Shea (July 2010). "Dual Coding with STDP in a Spiking Recurrent Neural Network Model of the Hippocampus". In: *PLOS Computational Biology* 6.7, e1000839. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1000839.

Buzsáki, György (Jan. 1989). "Two-stage model of memory trace formation: A role for "noisy" brain states". In: *Neuroscience* 31.3, pp. 551–570. ISSN: 0306-4522. DOI: 10.1016/0306-4522(89)90423-5.

Buzsáki, György (Jan. 2002). "Theta oscillations in the hippocampus". In: *Neuron* 33.3, pp. 325–340. DOI: 10.1016/s0896-6273(02)00586-x.

Buzsáki, György, Zsolt Horváth, Ronald Urioste, Jamille Hetke, and Kensall Wise (May 1992). "High-frequency network oscillation in the hippocampus". In: *Science* 256.5059, pp. 1025–1027. ISSN: 1095-9203. DOI: 10.1126/science.1589772.

Byrne, Patrick, Suzanna Becker, and Neil Burgess (Apr. 2007). "Remembering the past and imagining the future: A neural model of spatial memory and imagery". In: *Psychological Review* 114.2, pp. 340–375. DOI: 10.1037/0033-295x.114.2.340.

Carandini, Matteo and David J. Heeger (Nov. 2011). "Normalization as a canonical neural computation". In: *Nature Reviews Neuroscience* 13.1, pp. 51–62. ISSN: 1471-0048. DOI: 10.1038/nrn3136.

Carnevale, Federico, Victor de Lafuente, Ranulfo Romo, Omri Barak, and Néstor Parga (2015). "Dynamic control of response criterion in premotor cortex during perceptual detection under temporal uncertainty". In: *Neuron*. DOI: 10.1016/j.neuron.2015.04.014.

Carpenter, Francis, Daniel Manson, Kathryn J. Jeffery, Neil Burgess, and Caswell Barry (May 2015). "Grid Cells Form a Global Representation of Connected Environments". In: *Current Biology* 25.9, pp. 1176–1182. ISSN: 09609822. DOI: 10.1016/j.cub.2015.02.037.

Carr, Margaret F., Shantanu P. Jadhav, and Loren M. Frank (2011). "Hippocampal replay in the awake state: A potential substrate for memory consolidation and retrieval". In: *Nature neuroscience*. DOI: 10.1038/nn.2732.

Cashin-Garbutt, April and Sainsbury Wellcome Centre (June 2023). *TReND-CaMinA: Improving access to computational neuroscience and machine learning in Africa*. https://www.sainsburywellcome.org/web/blog/trend-camina-improving-access-computational-neuroscience-and-machine-learning-africa. Accessed: 2025-06-23. URL: https://www.sainsburywellcome.org/web/blog/trend-camina-improving-access-computational-neuroscience-and-machine-learning-africa.

Castillo, Jose del and Bernard Katz (1954). "Quantal components of the end-plate potential". In: *The Journal of physiology* 124.3, p. 560. DOI: 10.1113/jphysiol.1954.sp005129.

Cavanagh, J. F., C. M. Figueroa, M. X. Cohen, and M. J. Frank (Nov. 2011). "Frontal theta reflects uncertainty and unexpectedness during exploration and exploitation". In: *Cerebral Cortex* 22.11, pp. 2575–2586. DOI: 10.1093/cercor/bhr332.

Chadwick, Angus, Mark C. W. van Rossum, and Matthew F. Nolan (Feb. 2015). "Independent theta phase coding accounts for CA1 population sequences and enables flexible remapping". In: *eLife* 4. DOI: 10.7554/elife.03542.

Chen, Weiliang, Tristan Carel, Omar Awile, Nicola Cantarutti, Giacomo Castiglioni, Alessandro Cattabiani, Baudouin Del Marmol, Iain Hepburn, James G. King, Christos Kotsalos, Pramod Kumbhar, Jules Lallouette, Samuel Melchior, Felix Schürmann, and Erik De Schutter (Oct. 2022). "STEPS 4.0: Fast and memory-efficient molecular simulations of neurons at the nanoscale". In: *Frontiers in Neuroinformatics* 16. ISSN: 1662-5196. DOI: 10.3389/fninf.2022.883742.

Cheng, Sen and Loren M. Frank (Jan. 2008). "New experiences enhance coordinated neural activity in the hippocampus". In: *Neuron* 57.2, pp. 303–313. DOI: 10.1016/j.neuron.2007.11.035.

Chevalier-Boisvert, Maxime, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry (2023). "Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks". In: *CoRR* abs/2306.13831. DOI: 10.48550/arXiv.2306.13831.

Chowdhury, Raeed H., Joshua I. Glaser, and Lee E. Miller (2020). "Area 2 of primary somatosensory cortex encodes kinematics of the whole arm". In: *eLife*, e48198. DOI: 10.7554/elife.48198.

Chrobak, James J., Robert W. Stackman, and Thomas J. Walsh (1989). *Intraseptal administration of muscimol produces dose-dependent memory impairments in the rat*. DOI: 10.1016/S0163-1047(89)90472-X.

Chu, Tianhao, Zilong Ji, Junfeng Zuo, Yuanyuan Mi, Wen-Hao Zhang, Tiejun Huang, Daniel Bush, Neil Burgess, and Si Wu (2022). "Firing rate adaptation in continuous attractor neural networks accounts for theta phase shift of hippocampal place cells". In: *bioRxiv*. DOI: 10.7554/eLife.87055.2.

Colgin, Laura Lee, Tobias Denninger, Marianne Fyhn, Torkel Hafting, Tora Bonnevie, Ole Jensen, May-Britt Moser, and Edvard I. Moser (2009). "Frequency of gamma oscillations routes flow of information in the hippocampus". In: *Nature* 462.7271, pp. 353–357. DOI: 10.1038/nature08573.

Constantinescu, Alexandra O., Jill X. O'Reilly, and Timothy E. J. Behrens (June 2016). "Organizing conceptual knowledge in humans with a gridlike code". In: *Science* 352.6292, pp. 1464–1468. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aaf0941.

Cothi, William de and Caswell Barry (Dec. 2020a). "Neurobiological successor features for spatial navigation". In: *Hippocampus* 30.12, pp. 1347–1355. ISSN: 1050-9631, 1098-1063. DOI: 10.1002/hipo.23246.

Cothi, William de and Caswell Barry (2020b). "Neurobiological successor features for spatial navigation". In: *Hippocampus* 30.12, pp. 1347–1355. ISSN: 1098-1063. DOI: 10.1002/hipo.23246.

Cothi, William de, Nils Nyberg, Eva-Maria Griesbauer, Carole Ghanamé, Fiona Zisch, Julie M. Lefort, Lydia Fletcher, Coco Newton, Sophie Renaudineau, Daniel Bendor, Hugo J. Spiers, and Caswell Barry (2022a). "Predictive maps in rats and humans for spatial navigation". In: *Current Biology* 32.17, pp. 3676–3689. DOI: 10.1016/j.cub.2022.06.090.

Cothi, William de, Nils Nyberg, Eva-Maria Griesbauer, Carole Ghanamé, Fiona Zisch, Julie M. Lefort, Lydia Fletcher, Coco Newton, Sophie Renaudineau, Daniel Bendor, Hugo J. Spiers, and Caswell Barry (2022b). "Predictive maps in rats

and humans for spatial navigation". In: *Current Biology* 32.17, pp. 3676–3689. DOI: 10.1016/j.cub.2022.06.090.

Cueva, Christopher J. and Xue-Xin Wei (2018). *Emergence of grid-like representations by training recurrent neural networks to perform spatial localization*. DOI: 10.48550/arXiv.1803.07770.

Cunningham, John P. and Byron M. Yu (2014). "Dimensionality reduction for large-scale neural recordings". In: *Nature neuroscience*. DOI: 10.1038/nn.3776.

Dale, H. (1935). *Pharmacology and nerve-endings (Walter Ernest Dixon Memorial Lecture)*. DOI: 10.1136/bmj.2.3859.1161.

Daw, Nathaniel D, Yael Niv, and Peter Dayan (Nov. 2005). "Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control". In: *Nature Neuroscience* 8.12, pp. 1704–1711. ISSN: 1546-1726. DOI: 10.1038/nn1560.

Daw, Nathaniel D. (2012). "Model-based reinforcement learning as cognitive search: Neurocomputational theories". In: *Cognitive search: Evolution, algorithms and the brain*, pp. 195–208. URL: https://www.princeton.edu/~ndaw/d11a.pdf.

Dayan, Peter (July 1993). "Improving generalization for temporal difference learning: The successor representation". In: *Neural Computation* 5.4, pp. 613–624. DOI: 10.1162/neco.1993.5.4.613.

Dayan, Peter, Geoffrey E. Hinton, Radford M. Neal, and Richard S. Zemel (Sept. 1995). "The Helmholtz machine". In: *Neural Computation* 7.5, pp. 889–904. DOI: 10.1162/neco.1995.7.5.889.

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)*. DOI: 10.1111/j.2517-6161.1977.tb01600.x.

Deneve, S., P. E. Latham, and A. Pouget (Aug. 2001). "Efficient computation and cue integration with noisy population codes". In: *Nature Neuroscience* 4.8, pp. 826–831. ISSN: 1546-1726. DOI: 10.1038/90541.

Denovellis, Eric L., Anna K. Gillespie, Michael E. Coulter, Marielena Sosa, Jason E. Chung, Uri T. Eden, and Loren M. Frank (2021). "Hippocampal replay of

experience at real-world speeds". In: *eLife*, e64505. DOI: 10.7554/elife.64505.sa2.

Dominé, Clémentine C. J., Rodrigo Carrasco-Davis, Luke Hollingsworth, Nikoloz Sirmpilatze, Adam L. Tyson, Devon Jarvis, Caswell Barry, and Andrew M. Saxe (Mar. 2024). "NeuralPlayground: A Standardised Environment for Evaluating Models of Hippocampus and Entorhinal Cortex". In: DOI: 10.1101/2024.03.06.583699.

Dong, Can, Antoine D. Madar, and Mark E. J. Sheffield (May 2021). "Distinct place cell dynamics in CA1 and CA3 encode experience in new environments". In: *Nature Communications* 12.1, p. 2977. ISSN: 2041-1723. DOI: 10.1038/s41467-021-23260-3.

Dong, Zhe, Bryan Seybold, Kevin Murphy, and Hung Bui (2020). "Collapsed amortized variational inference for switching nonlinear dynamical systems". In: *International Conference on Machine Learning*. PMLR, pp. 2638–2647. DOI: 10.48550/arXiv.1910.09588.

Dorrell, William, Peter E. Latham, Timothy E. J. Behrens, and James C. R. Whittington (2023). *Actionable neural representations: Grid cells from minimal constraints*. DOI: 10.48550/arXiv.2209.15563.

Doya, Kenji (Jan. 2000). "Reinforcement learning in continuous time and space". In: *Neural Computation* 12.1, pp. 219–245. DOI: 10.1162/089976600300015961.

Drieu, Céline and Michaël Zugaro (2019). "Hippocampal sequences during exploration: Mechanisms and functions". In: *Frontiers in Cellular Neuroscience* 13, p. 232. ISSN: 1662-5102. DOI: 10.3389/fncel.2019.00232.

Duncker, Lea, Gergo Bohner, Julien Boussard, and Maneesh Sahani (2019). "Learning interpretable continuous-time models of latent stochastic dynamical systems". In: *International conference on machine learning*. PMLR. DOI: 10.48550/arXiv.1902.04420.

Dupret, David, Joseph O'Neill, Barty Pleydell-Bouverie, and Jozsef Csicsvari (July 2010). "The reorganization and reactivation of hippocampal maps predict spatial memory performance". In: *Nature Neuroscience* 13.8, pp. 995–1002. DOI: 10.1038/nn.2599.

Eliav, Tamir, Shir R. Maimon, Johnatan Aljadeff, Misha Tsodyks, Gily Ginosar, Liora Las, and Nachum Ulanovsky (May 2021). "Multiscale representation of very large environments in the hippocampus of flying bats". In: *Science* 372.6545, eabg4020. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.abg4020.

Epstein, Joshua M. (2008). "Why Model?" In: *Journal of Artificial Societies and Social Simulation* 11.4, p. 12. ISSN: 1460-7425. URL: https://www.jasss.org/11/4/12.html.

Erdem, Uğur M. and Michael E. Hasselmo (2012). "A goal-directed spatial navigation model using forward trajectory planning based on grid cells". In: *European Journal of Neuroscience* 35.6, pp. 916–931. ISSN: 1460-9568. DOI: 10.1111/j.1460-9568.2012.08015.x.

Ergorul, Ceren and Howard B. Eichenbaum (Apr. 2006). "Essential role of the hippocampal formation in rapid learning of higher-order sequential associations". In: *The Journal of Neuroscience* 26.15, pp. 4111–4117. DOI: 10.1523/jneurosci.0441-06.2006.

Fang, Ching, Dmitriy Aronov, L. F. Abbott, and Emily L. Mackevicius (2022). "Neural learning rules for generating flexible predictions and computing the successor representation". In: *bioRxiv*. DOI: 10.1101/2022.05.18.492543.

Fang, Ching, Dmitriy Aronov, L. F. Abbott, and Emily L. Mackevicius (2023). "Neural learning rules for generating flexible predictions and computing the successor representation". In: *eLife*, e80680. DOI: 10.7554/elife.80680.

Feng, Ting, Delia Silva, and David J. Foster (2015). "Dissociation between the experience-dependent development of hippocampal theta sequences and single-trial phase precession". In: *The Journal of Neuroscience* 35.12, pp. 4890–4902. DOI: 10.1523/jneurosci.2614-14.2015.

Fenton, André A. and Robert U. Muller (1998). "Place cell discharge is extremely variable during individual passes of the rat through the firing field". In: *Proceedings of the National Academy of Sciences*. DOI: 10.1073/pnas.95.6.3182.

Fisher, Ronald Aylmer (1925). "Theory of statistical estimation". In: *Mathematical proceedings of the Cambridge philosophical society*. Cambridge University Press. DOI: 10.1017/S0305004100009580.

Foster, David J. and Matthew A. Wilson (2007). "Hippocampal theta sequences". In: *Hippocampus* 17.11, pp. 1093–1099. DOI: 10.1002/hipo.20345.

Fries, Pascal (Oct. 2015). "Rhythms for cognition: Communication through coherence". In: *Neuron* 88.1, pp. 220–235. DOI: 10.1016/j.neuron.2015.09.034.

Friston, Karl J. (Jan. 2010). "The free-energy principle: A unified brain theory?" In: *Nature Reviews Neuroscience* 11.2, pp. 127–138. DOI: 10.1038/nrn2787.

Fuhs, Mark C. and David S. Touretzky (2006). "A spin glass model of path integration in rat medial entorhinal cortex". In: *The Journal of Neuroscience* 26.16, pp. 4266–4276. ISSN: 0270-6474. DOI: 10.1523/jneurosci.4353-05.2006.

Fyhn, Marianne, Torkel Hafting, Alessandro Treves, May-Britt Moser, and Edvard I. Moser (Feb. 2007). "Hippocampal remapping and grid realignment in entorhinal cortex". In: *Nature* 446.7132, pp. 190–194. DOI: 10.1038/nature05601.

El-Gaby, Mohamady, Adam Loyd Harris, James C. R. Whittington, William Dorrell, Arya Bhomick, Mark E. Walton, Thomas Akam, and Timothy E. J. Behrens (Nov. 2024). "A cellular basis for mapping behavioural structure". In: *Nature* 636.8043, pp. 671–680. ISSN: 1476-4687. DOI: 10.1038/s41586-024-08145-x.

Gao, Yuanjun, Evan W. Archer, Liam Paninski, and John P. Cunningham (2016). "Linear dynamical neural population models through nonlinear embeddings". In: *Advances in Neural Information Processing Systems*. DOI: 10.48550/arXiv.1605.08454.

Garvert, Mona M., Raymond J. Dolan, and Timothy E. J. Behrens (Apr. 2017). "A map of abstract relational knowledge in the human hippocampal–entorhinal cortex". In: *eLife* 6, e17086. ISSN: 2050-084X. DOI: 10.7554/eLife.17086.

Gauthier, Jeffrey L. and David W. Tank (July 2018). "A dedicated population for reward coding in the hippocampus". In: *Neuron* 99.1, 179–193.e7. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2018.06.008.

Geerts, Jesse P., Fabian Chersi, Kimberly L. Stachenfeld, and Neil Burgess (Dec. 2020). "A general model of hippocampal and dorsal striatal learning and decision making". In: *Proceedings of the National Academy of Sciences* 117.49, pp. 31427–31437. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.2007981117.

George, Dileep, Rajeev V. Rikhye, Nishad Gothoskar, J. Swaroop Guntupalli, Antoine Dedieu, and Miguel Lázaro-Gredilla (Dec. 2021). "Clone-structured graph representations enable flexible learning and vicarious evaluation of cognitive maps". In: *Nature Communications* 12.1, p. 2392. ISSN: 2041-1723. DOI: 10.1038/s41467-021-22559-5.

George, Tom M. (2023). "Theta sequences as eligibility traces: A biological solution to credit assignment". In: *International Conference on Learning Representations 2023 (TinyPapers track)*. DOI: 10.48550/arXiv.2305.08124.

George, Tom M., William de Cothi, Kimberly L. Stachenfeld, and Caswell Barry (Mar. 2023a). "Rapid learning of predictive maps with STDP and theta phase precession". In: *eLife* 12. DOI: 10.7554/elife.80663.

George, Tom M., Pierre Glaser, Kimberly L. Stachenfeld, Caswell Barry, and Claudia Clopath (2025). "SIMPL: Scalable and hassle-free optimisation of neural representations from behaviour". In: *The Thirteenth International Conference on Learning Representations*. DOI: 10.1101/2024.11.11.623030.

George, Tom M. and Pietro Lio (Nov. 2019). "Unsupervised machine learning for data encoding applied to ovarian cancer transcriptomes". In: DOI: 10.1101/855593.

George, Tom M., Mehul Rastogi, William de Cothi, Claudia Clopath, Kimberly L. Stachenfeld, and Caswell Barry (2024). "RatInABox, a toolkit for modelling locomotion and neuronal activity in continuous environments". In: *eLife*. DOI: 10.7554/elife.85274.

George, Tom M., Kimberly L. Stachenfeld, Caswell Barry, Claudia Clopath, and Tomoki Fukai (2023b). "A generative model of the hippocampal formation trained with theta driven local learning rules". In: *Thirty-seventh Conference on Neural Information Processing Systems*. DOI: 10.1101/2023.12.12.571268.

Georgopoulos, Apostolos P., Andrew B. Schwartz, and Ronald E. Kettner (1986). "Neuronal population coding of movement direction". In: *Science* 233.4771, pp. 1416–1419. DOI: 10.1126/science.3749885.

Gershman, Samuel J. (Aug. 2018). "The Successor Representation: Its Computational Logic and Neural Substrates". In: *The Journal of Neuroscience* 38.33, pp. 7193–7200. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.0151-18.2018.

Gershman, Samuel J. (Sept. 2019). "The generative adversarial brain". In: *Frontiers in Artificial Intelligence* 2. DOI: 10.3389/frai.2019.00018.

Glaser, Joshua I., Ari S. Benjamin, Raeed H. Chowdhury, Matthew G. Perich, Lee E. Miller, and Konrad Paul Kording (2020). "Machine learning for neural decoding". In: *eNeuro* 4. DOI: 10.1523/ENEURO.0506-19.2020.

Gondur, Rabia, Usama Bin Sikandar, Evan Schaffer, Mikio Christian Aoi, and Stephen L. Keeley (2023). "Multi-modal Gaussian process variational autoencoders for neural and behavioral data". In: *arXiv preprint arXiv:2310.03111*. DOI: 10.48550/arXiv.2310.03111.

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). *Generative adversarial networks*. DOI: 10.48550/arXiv.1406.2661.

Goodman, Dan (2008). "Brian: a simulator for spiking neural networks in Python". In: *Frontiers in Neuroinformatics* 2. ISSN: 1662-5196. DOI: 10.3389/neuro.11.005.2008.

Green, John D. and Arnaldo A. Arduini (Nov. 1954). "Hippocampal electrical activity in arousal". In: *Journal of Neurophysiology* 17.6, pp. 533–557. ISSN: 1522-1598. DOI: 10.1152/jn.1954.17.6.533.

Guerguiev, Jordan, Timothy P. Lillicrap, and Blake A. Richards (Dec. 2017). "Towards deep learning with segregated dendrites". In: *eLife* 6. DOI: 10.7554/elife.22901.

Gustafson, Nicholas J. and Nathaniel D. Daw (Oct. 2011). "Grid cells, place cells, and geodesic generalization for spatial reinforcement learning". In: *PLOS Computational Biology* 7.10, e1002235. DOI: 10.1371/journal.pcbi.1002235.

Györfi, László, Michael Kohler, Adam Krzyzak, and Harro Walk (2006). *A distribution-free theory of nonparametric regression*. Springer Science & Business Media.

Hafting, Torkel, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I. Moser (June 2005). "Microstructure of a spatial map in the entorhinal cortex". In: *Nature* 436.7052, pp. 801–806. DOI: 10.1038/nature03721.

Han, Dongqi, Kenji Doya, and Jun Tani (Sept. 2020). "Self-organization of action hierarchy and compositionality by reinforcement learning with recurrent neural

networks". In: *Neural Networks* 129, pp. 149–162. DOI: 10.1016/j.neunet.2020.06.002.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant (Sept. 2020). "Array programming with NumPy". In: *Nature* 585.7825, pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

Harris, Kenneth D. (May 2005). "Neural signatures of cell assembly organization". In: *Nature Reviews Neuroscience* 6.5, pp. 399–407. DOI: 10.1038/nrn1669.

Hartley, Tom, Neil Burgess, Colin Lever, Francesca Cacucci, and John O'Keefe (2000). "Modeling place fields in terms of the cortical inputs to the hippocampus". In: *Hippocampus* 10.4, pp. 369–379. ISSN: 1098-1063. DOI: 10.1002/1098-1063(2000)10:4<369::aid-hipo3>3.0.co;2-0.

Hartley, Tom, Iris Trinkler, and Neil Burgess (2004). "Geometric determinants of human spatial memory". In: *Cognition* 94.1, pp. 39–75. DOI: 10.1016/j.cognition.2003.12.001.

Harvey, Christopher D., Philip Coen, and David W. Tank (2012). "Choice-specific sequences in parietal cortex during a virtual-navigation decision task". In: *Nature*. DOI: 10.1038/nature10918.

Hassabis, Demis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick (July 2017). "Neuroscience-Inspired Artificial Intelligence". In: *Neuron* 95.2, pp. 245–258. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2017.06.011.

Hasselmo, Michael E., Clara Bodelón, and Bradley P. Wyble (Apr. 2002). "A Proposed Function for Hippocampal Theta Rhythm: Separate Phases of Encoding and Retrieval Enhance Reversal of Prior Learning". In: *Neural Computation* 14.4, pp. 793–817. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/089976602317318965.

Hasselmo, Michael E. and Howard B. Eichenbaum (2005). "Hippocampal mechanisms for the context-dependent retrieval of episodes". In: *Neural networks* 18.9, pp. 1172–1190. DOI: 10.1016/j.neunet.2005.08.007.

Hasselmo, Michael E. and Chantal E. Stern (Jan. 2014). "Theta rhythm and the encoding and retrieval of space and time". In: *NeuroImage* 85, pp. 656–666. DOI: 10.1016/j.neuroimage.2013.06.022.

Hebb, Donald Olding (1949). *The Organization of Behavior: A Neuropsychological Theory*. New York: Wiley & Sons.

Hepburn, Iain, Weiliang Chen, Stefan Wils, and Erik De Schutter (May 2012). "STEPS: Efficient simulation of stochastic reaction–diffusion models in realistic morphologies". In: *BMC Systems Biology* 6.1. ISSN: 1752-0509. DOI: 10.1186/1752-0509-6-36.

Hernandez, Daniel, Antonio Khalil Moretti, Ziqiang Wei, Shreya Saxena, John P. Cunningham, and Liam Paninski (2018). "Nonlinear evolution via spatially-dependent linear dynamics for electrophysiology and calcium data". In: *arXiv preprint arXiv:1811.02459*. DOI: 10.48550/arXiv.1811.02459.

Hines, M. L. and N. T. Carnevale (Aug. 1997). "The NEURON simulation environment". In: *Neural Computation* 9.6, pp. 1179–1209. ISSN: 1530-888X. DOI: 10.1162/neco.1997.9.6.1179.

Hinton, Geoffrey E., Peter Dayan, Brendan J. Frey, and Radford M. Neal (May 1995). "The "wake-sleep" algorithm for unsupervised neural networks". In: *Science* 268.5214, pp. 1158–1161. DOI: 10.1126/science.7761831.

Hodgkin, Alan L. and Andrew F. Huxley (1952). "A quantitative description of membrane current and its application to conduction and excitation in nerve". In: *The Journal of physiology* 117.4, p. 500. DOI: 10.1113/jphysiol.1952.sp004764.

Hollup, Stig A., Sturla Molden, James G. Donnett, May-Britt Moser, and Edvard I. Moser (Mar. 2001). "Accumulation of Hippocampal Place Fields at the Goal Location in an Annular Watermaze Task". In: *The Journal of Neuroscience* 21.5, pp. 1635–1644. ISSN: 1529-2401. DOI: 10.1523/jneurosci.21-05-01635.2001.

Holscher, Christian, Roger Anwyl, and Michael J. Rowan (Aug. 1997). "Stimulation on the positive phase of hippocampal theta rhythm induces long-term potentiation that can be depotentiated by stimulation on the negative phase in area CA1". In: *The Journal of Neuroscience* 17.16, pp. 6470–6477. DOI: 10.1523/jneurosci.17-16-06470.1997.

Hopfield, John J. (Apr. 1982). "Neural networks and physical systems with emergent collective computational abilities." In: *Proceedings of the National Academy of Sciences* 79.8, pp. 2554–2558. ISSN: 1091-6490. DOI: 10.1073/pnas.79.8.2554.

Høydal, Øyvind Arne, Emilie Ranheim Skytøen, Sebastian Ola Andersson, May-Britt Moser, and Edvard I. Moser (2019). "Object-vector coding in the medial entorhinal cortex". In: *Nature*. DOI: 10.1038/s41586-019-1077-7.

Hu, Amber, David Zoltowski, Aditya Nair, David Anderson, Lea Duncker, and Scott W. Linderman (2024). "Modeling latent neural dynamics with Gaussian process switching linear dynamical systems". In: *arXiv preprint arXiv:2408.03330*. DOI: 10.48550/arXiv.2408.03330.

Hurwitz, Cole, Akash Srivastava, Kai Xu, Justin Jude, Matthew Perich, Lee E. Miller, and Matthias Hennig (2021). "Targeted neural dynamical modeling". In: *Advances in Neural Information Processing Systems* 34, pp. 29379–29392. DOI: 10.48550/arXiv.2110.14853.

Huxter, John R., Neil Burgess, and John O'Keefe (Oct. 2003). "Independent rate and temporal coding in hippocampal pyramidal cells". In: *Nature* 425.6960, pp. 828–832. DOI: 10.1038/nature02058.

Hyvärinen, Aapo and Petteri Pajunen (1999). "Nonlinear independent component analysis: Existence and uniqueness results". In: *Neural networks*. DOI: 10.1016/s0893-6080(98)00140-3.

Jeewajee, Ali, Caswell Barry, Vincent Douchamps, Daniel Manson, Colin Lever, and Neil Burgess (Feb. 2014). "Theta phase precession of grid and place cell firing in open environments". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 369.1635, p. 20120532. ISSN: 0962-8436, 1471-2970. DOI: 10.1098/rstb.2012.0532.

Jensen, Kristopher, Ta-Chu Kao, Marco Tripodi, and Guillaume Hennequin (2020). "Manifold GPLVMs for discovering non-Euclidean latent structure in neural data". In: *Advances in Neural Information Processing Systems*. DOI: 10.48550/arXiv.2006.07429.

Jensen, Ole and John E. Lisman (1996). "Hippocampal CA3 region predicts memory sequences: Accounting for the phase precession of place cells". In: *Learning & memory* 3.2-3, pp. 279–287. DOI: 10.1101/lm.3.2-3.279.

Joensen, Bardhur H., Daniel Bush, Umesh Vivekananda, Aidan J. Horner, James A. Bisby, Beate Diehl, Anna Miserocchi, Andrew W. McEvoy, Matthew C. Walker, and Neil Burgess (May 2023). "Hippocampal theta activity during encoding promotes subsequent associative memory in humans". In: *Cerebral Cortex*. DOI: 10.1093/cercor/bhad162.

Johnson, Adam and A. David Redish (Nov. 2007). "Neural Ensembles in CA3 Transiently Encode Paths Forward of the Animal at a Decision Point". In: *The Journal of Neuroscience* 27.45, pp. 12176–12189. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.3761-07.2007.

Juliani, Arthur, Samuel Barnett, Brandon Davis, Margaret Sereno, and Ida Momennejad (2022). "Neuro-Nav: A library for neurally-plausible reinforcement learning". In: *arXiv*. DOI: 10.32470/ccn.2022.1212-0.

Jun, James J., Nicholas A. Steinmetz, Joshua H. Siegle, Daniel J. Denman, Marius Bauza, Brian Barbarits, Albert K. Lee, Costas A. Anastassiou, Alexandru Andrei, Çağatay Aydın, Maja Barbic, Timothy J. Blanche, Vincent Bonin, Joana Couto, Barundeb Dutta, Sergey L. Gratiy, Diego A. Gutnisky, Michael Häusser, Bill Karsh, Peter Ledochowitsch, Carolina M. Lopez, Partha Mitra, Jeroen D. Mussche, Michael Okun, Marius Pachitariu, Jan Putzeys, Peter D. Rich, Cyrille Rossant, Wei-Lian Sun, Karel Svoboda, Adam Tring, Marleen Welkenhuysen, and Timothy D. Harris (2017). "Fully integrated silicon probes for high-density recording of neural activity". In: *Nature*.

Kalman, Rudolph Emil (1960). "A new approach to linear filtering and prediction problems". In: DOI: 10.1115/1.3662552.

Kay, Kenneth, Jason E. Chung, Marielena Sosa, Jonathan S. Schor, Mattias P. Karlsson, Margaret C. Larkin, Daniel F. Liu, and Loren M. Frank (Feb. 2020). "Constant sub-second cycling between representations of possible futures in the hippocampus". In: *Cell* 180.3, 552–567.e25. DOI: 10.1016/j.cell.2020.01.014.

Khona, Mikail and Ila R. Fiete (2021). "Attractor and integrator networks in the brain". In: *arXiv*. DOI: 10.48550/arxiv.2112.03978.

Kim, Timothy D., Thomas Z. Luo, Jonathan W. Pillow, and Carlos D. Brody (2021). "Inferring latent dynamics underlying neural population activity via

neural differential equations". In: *International Conference on Machine Learning*. PMLR, pp. 5551–5561.

Kingma, Diederik P. and Max Welling (2014). "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations*.

Kingma, Diederik P. and Max Welling (2022). *Auto-encoding variational Bayes*. DOI: 10.48550/arXiv.1312.6114.

Kirkpatrick, James, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell (Mar. 2017). "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the National Academy of Sciences* 114.13, pp. 3521–3526. ISSN: 1091-6490. DOI: 10.1073/pnas.1611835114.

Kjelstrup, Kirsten Brun, Trygve Solstad, Vegard Heimly Brun, Torkel Hafting, Stefan Leutgeb, Menno P. Witter, Edvard I. Moser, and May-Britt Moser (July 2008). "Finite scale of spatial representation in the hippocampus". In: *Science* 321.5885, pp. 140–143. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1157086.

Knill, David C. and Alexandre Pouget (Dec. 2004). "The Bayesian brain: the role of uncertainty in neural coding and computation". In: *Trends in Neurosciences* 27.12, pp. 712–719. ISSN: 0166-2236. DOI: 10.1016/j.tins.2004.10.007.

Kobak, Dmitry, Wieland Brendel, Christos Constantinidis, Claudia E. Feierstein, Adam Kepecs, Zachary F. Mainen, Xue-Lian Qi, Ranulfo Romo, Naoshige Uchida, and Christian K. Machens (2016). "Demixed principal component analysis of neural population data". In: *eLife*. DOI: 10.7554/elife.10989.022.

Koene, Randal A., Anatoli Gorchetchnikov, Robert C. Cannon, and Michael E. Hasselmo (2003). "Modeling goal-directed spatial navigation in the rat based on physiological data from the hippocampal formation". In: *Neural Networks* 16.5-6, pp. 577–584. DOI: 10.1016/s0893-6080(03)00106-0.

Kording, Konrad Paul and Peter Konig (2001). "Supervised and unsupervised learning with two sites of synaptic integration". In: *Journal of Computational Neuroscience* 11.3, pp. 207–215. DOI: 10.1023/a:1013776130161.

Krishnan, Rahul G., Uri Shalit, and David Sontag (2015). "Deep kalman filters". In: *arXiv preprint arXiv:1511.05121*. DOI: 10.48550/arXiv.1511.05121.

Kropff, Emilio, James E. Carmichael, May-Britt Moser, and Edvard I. Moser (2015). "Speed cells in the medial entorhinal cortex". In: *Nature* 523.7561, pp. 419–424. ISSN: 0028-0836. DOI: 10.1038/nature14622.

Larkum, Matthew E. (May 2022). "Are dendrites conceptually useful?" In: *Neuroscience* 489, pp. 4–14. DOI: 10.1016/j.neuroscience.2022.03.008.

Lawrence, Neil (2003). "Gaussian process latent variable models for visualisation of high dimensional data". In: *Advances in neural information processing systems* 16. URL: https://proceedings.neurips.cc/paper/2003/file/9657c1fffd38824e5ab0472e022e577e-Paper.pdf.

Lee, J. Quinn, Alexandra T. Keinath, Erica Cianfarano, and Mark P. Brandon (Oct. 2023). "Identifying representational structure in CA1 to benchmark theoretical models of cognitive mapping". In: DOI: 10.1101/2023.10.08.561112.

Lever, Colin, Stephen Burton, Ali Jeewajee, John O'Keefe, and Neil Burgess (2009). "Boundary vector cells in the subiculum of the hippocampal formation". In: *The Journal of Neuroscience*. DOI: 10.1523/jneurosci.1319-09.2009.

Li, Kwan Tung, Junhao Liang, and Changsong Zhou (Jan. 2021). "Gamma oscillations facilitate effective learning in excitatory-inhibitory balanced neural circuits". In: *Neural Plasticity* 2021, pp. 1–18. DOI: 10.1155/2021/6668175.

Lieder, Itay, Vincent Adam, Or Frenkel, Sagi Jaffe-Dax, Maneesh Sahani, and Merav Ahissar (2019). "Perceptual bias reveals slow-updating in autism and fast-forgetting in dyslexia". In: *Nature neuroscience*. DOI: 10.1038/s41593-018-0308-9.

Lillicrap, Timothy P., Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey E. Hinton (June 2020). "Backpropagation and the brain". In: *Nature Reviews Neuroscience* 21.6, pp. 335–346. ISSN: 1471-003X, 1471-0048. DOI: 10.1038/s41583-020-0277-3.

Linderman, Scott W., Andrew C. Miller, Ryan P. Adams, David M. Blei, Liam Paninski, and Matthew J. Johnson (2016). "Recurrent switching linear dynamical systems". In: *arXiv preprint arXiv:1610.08466*. DOI: 10.48550/arXiv.1610.08466.

Lisman, John E. and Anthony A. Grace (June 2005). "The hippocampal-VTA loop: Controlling the entry of information into long-term memory". In: *Neuron* 46.5, pp. 703–713. DOI: 10.1016/j.neuron.2005.05.002.

Locatello, Francesco, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem (2019). "Challenging common assumptions in the unsupervised learning of disentangled representations". In: *International Conference on Machine Learning*. DOI: 10.48550/arXiv.1811.12359.

Low, Ryan J., Sam Lewallen, Dmitriy Aronov, Rhino Nevers, and David W. Tank (2018). "Probing variability in a cognitive map using manifold inference from neural dynamics". In: *BioRxiv*. DOI: 10.1101/418939.

Lubenov, Evgueniy V. and Athanassios G. Siapas (2009). "Hippocampal theta oscillations are travelling waves". In: *Nature* 459.7246, pp. 534–539. DOI: 10.1038/nature08010.

Maaten, Laurens van der and Geoffrey E. Hinton (2008). "Visualizing data using t-SNE". In: *Journal of Machine Learning Research*. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

MacKay, David J. C. (2003). *Information theory, inference, and learning algorithms*. Cambridge University Press. DOI: 10.1109/tit.2004.834752.

Macke, Jakob H., Lars Buesing, John P. Cunningham, Byron M. Yu, Krishna V. Shenoy, and Maneesh Sahani (2011). "Empirical models of spiking in neural populations". In: *Advances in Neural Information Processing Systems*. URL: https://papers.nips.cc/paper_files/paper/2011/hash/7143d7fbadfa4693b9eec507d9d37443-Abstract.html.

Mackevicius, Emily L., Andrew H. Bahle, Alex H. Williams, Shijie Gu, Natalia I. Denisenko, Mark S. Goldman, and Michale S. Fee (Feb. 2019). "Unsupervised discovery of temporal sequences in high-dimensional datasets, with applications to neuroscience". In: *eLife* 8, e38471. ISSN: 2050-084X. DOI: 10.7554/eLife.38471.

Mangion, Andrew Zammit, Ke Yuan, Visakan Kadirkamanathan, Mahesan Niranjan, and Guido Sanguinetti (2011). "Online variational inference for state-space

models with point-process observations". In: *Neural Computation*. DOI: 10.1162/neco_a_00156.

Mante, Valerio, David Sussillo, Krishna V. Shenoy, and William T. Newsome (2013). "Context-dependent computation by recurrent dynamics in prefrontal cortex". In: *Nature* 503.7474, pp. 78–84. ISSN: 0028-0836. DOI: 10.1038/nature12742.

Markram, Henry, Joachim Lübke, Michael Frotscher, and Bert Sakmann (Jan. 1997). "Regulation of Synaptic Efficacy by Coincidence of Postsynaptic APs and EPSPs". In: *Science* 275.5297, pp. 213–215. ISSN: 1095-9203. DOI: 10.1126/science.275.5297.213.

Marr, David (1971). "Simple memory: A theory for archicortex". In: *Philosophical Transactions of the Royal Society of London. B, Biological Sciences* 262.841, pp. 23–81. DOI: 10.1098/rstb.1971.0012.

Marr, David (1982). *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. USA: Henry Holt and Co., Inc. ISBN: 0716715678. DOI: 10.7551/mitpress/9780262514620.001.0001.

Mathis, Alexander, Pranav Mamidanna, Kevin M. Cury, Taiga Abe, Venkatesh N. Murthy, Mackenzie W. Mathis, and Matthias Bethge (2018). "DeepLabCut: Markerless pose estimation of user-defined body parts with deep learning". In: *Nature Neuroscience*. DOI: 10.1038/s41593-018-0209-y.

Mattar, Marcelo G. and Nathaniel D. Daw (Oct. 2018). "Prioritized memory access explains planning and hippocampal replay". In: *Nature Neuroscience* 21.11, pp. 1609–1617. DOI: 10.1038/s41593-018-0232-z.

Maurer, Andrew P., Stephen L. Cowen, Sara N. Burke, Carol A. Barnes, and Bruce L. McNaughton (2006). "Organization of hippocampal cell assemblies based on theta phase precession". In: *Hippocampus* 16.9, pp. 785–794. ISSN: 1050-9631. DOI: 10.1002/hipo.20202.

McInnes, Leland, John Healy, and James Melville (2018). "Umap: Uniform manifold approximation and projection for dimension reduction". In: *arXiv preprint arXiv:1802.03426*. DOI: 10.21105/joss.00861.

McNaughton, Bruce L., Carol A. Barnes, J. L. Gerrard, K. Gothard, M. W. Jung, J. J. Knierim, H. Kudrimoti, Y. Qin, William E. Skaggs, M. Suster, and K. L. Weaver (Jan. 1996). "Deciphering the hippocampal polyglot: The hippocampus as a path

integration system". In: *Journal of Experimental Biology* 199.1, pp. 173–185. DOI: 10.1242/jeb.199.1.173.

McNaughton, Bruce L., Carol A. Barnes, and John O'Keefe (1983). "The contributions of position, direction, and velocity to single unit activity in the hippocampus of freely-moving rats". In: *Experimental brain research*. DOI: 10.1007/bf00237147.

McNaughton, Bruce L., Francesco P. Battaglia, Ole Jensen, Edvard I. Moser, and May-Britt Moser (2006). "Path integration and the neural basis of the 'cognitive map'". In: *Nature Reviews Neuroscience* 7.8, pp. 663–678. ISSN: 1471-003X. DOI: 10.1038/nrn1932.

Mehta, Mayank R. (Dec. 2001). "Neuronal dynamics of predictive coding". In: *The Neuroscientist* 7.6, pp. 490–495. DOI: 10.1177/107385840100700605.

Mehta, Mayank R., Carol A. Barnes, and Bruce L. McNaughton (1997). "Experience-dependent, asymmetric expansion of hippocampal place fields". In: *Proceedings of the National Academy of Sciences*. DOI: 10.1073/pnas.94.16.8918.

Mehta, Mayank R., Albert K. Lee, and Matthew A. Wilson (June 2002). "Role of experience and oscillations in transforming a rate code into a temporal code". In: *Nature* 417.6890, pp. 741–746. DOI: 10.1038/nature00807.

Mehta, Mayank R., Michael C. Quirk, and Matthew A. Wilson (2000). "Experience-dependent asymmetric shape of hippocampal receptive fields". In: *Neuron* 25, pp. 707–715. DOI: 10.1016/s0896-6273(00)81072-7.

Merel, Josh, Diego Aldarondo, Jesse Marshall, Yuval Tassa, Greg Wayne, and Bence P. Ölveczky (2019). *Deep neuroethology of a virtual rodent*. DOI: 10.48550/ARXIV.1911.09451.

Mizuseki, Kenji, Sebastien Royer, Kamran Diba, and György Buzsáki (Feb. 2012). "Activity dynamics and behavioral correlates of CA3 and CA1 hippocampal pyramidal neurons". In: *Hippocampus* 22.8, pp. 1659–1680. DOI: 10.1002/hipo.22002.

Mizuseki, Kenji, Anton Sirota, Eva Pastalkova, and György Buzsáki (Oct. 2009). "Theta oscillations provide temporal windows for local circuit computation in the entorhinal-hippocampal loop". In: *Neuron* 64.2, pp. 267–280. DOI: 10.1016/j.neuron.2009.08.037.

Momennejad, Ida and Marc W. Howard (2018). "Predicting the future with multi-scale successor representations". In: *bioRxiv*. DOI: 10.1101/449470.

Momennejad, Ida, Evan M. Russek, J. H. Cheong, Matthew M. Botvinick, Nathaniel D. Daw, and Samuel J. Gershman (Sept. 2017). "The successor representation in human reinforcement learning". In: *Nature Human Behaviour* 1.9, pp. 680–692. ISSN: 2397-3374. DOI: 10.1038/s41562-017-0180-8.

Morris, Richard G. M., P. Garrud, J. N. P. Rawlins, and John O'Keefe (June 1982). "Place navigation impaired in rats with hippocampal lesions". In: *Nature* 297.5868, pp. 681–683. ISSN: 1476-4687. DOI: 10.1038/297681a0.

Moser, Edvard I., Emilio Kropff, and May-Britt Moser (July 2008). "Place Cells, Grid Cells, and the Brain's Spatial Representation System". In: *Annual Review of Neuroscience* 31.1, pp. 69–89. ISSN: 1545-4126. DOI: 10.1146/annurev.neuro.31.061307.090723.

Moser, Edvard I., May-Britt Moser, and Bruce L. McNaughton (Nov. 2017). "Spatial representation in the hippocampal formation: A history". In: *Nature Neuroscience* 20.11, pp. 1448–1464. DOI: 10.1038/nn.4653.

Muller, Robert U. and John L. Kubie (1989). "The firing of hippocampal place cells predicts the future position of freely moving rats". In: *The Journal of Neuroscience*. DOI: 10.1523/jneurosci.09-12-04101.1989.

Muller, Robert U., John L. Kubie, and Russ Saypoff (July 1991). "The hippocampus as a cognitive graph (abridged version)". In: *Hippocampus* 1.3, pp. 243–246. DOI: 10.1002/hipo.450010306.

Nabyonga-Orem, Juliet, James Avoka Asamani, and Olu Olushayo (Mar. 2024). "Why Are African Researchers Left Behind in Global Scientific Publications? – A Viewpoint". In: *International Journal of Health Policy and Management*. ISSN: 2322-5939. DOI: 10.34172/ijhpm.2024.8149.

Nádasdy, Zoltán, Hajime Hirase, András Czurkó, Jozsef Csicsvari, and György Buzsáki (Nov. 1999). "Replay and Time Compression of Recurring Spike Sequences in the Hippocampus". In: *The Journal of Neuroscience* 19.21, pp. 9497–9507. ISSN: 1529-2401. DOI: 10.1523/jneurosci.19-21-09497.1999.

Nadel, Lynn and Morris Moscovitch (Apr. 1997). "Memory consolidation, retrograde amnesia and the hippocampal complex". In: *Current Opinion in Neurobiology* 7.2, pp. 217–227. ISSN: 0959-4388. DOI: 10.1016/s0959-4388(97)80010-4.

Nam, Hooram (2015). "Poisson extension of gaussian process factor analysis for modeling spiking neural populations". In: *Master's thesis, Department of Neural Computation and Behaviour, Max Planck Institute for Biological Cybernetics, Tübingen*.

O'Keefe, John (Jan. 1976). "Place units in the hippocampus of the freely moving rat". In: *Experimental Neurology* 51.1, pp. 78–109. DOI: 10.1016/0014-4886(76)90055-8.

O'Keefe, John and Neil Burgess (1996). "Geometric determinants of the place fields of hippocampal neurons". In: *Nature* 381.6581, pp. 425–428. ISSN: 0028-0836. DOI: 10.1038/381425a0.

O'Keefe, John and Jonathan Dostrovsky (1971). "The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely-moving rat". In: *Brain research*. DOI: 10.1016/0006-8993(71)90358-1.

O'Keefe, John and Lynn Nadel (1978). *The hippocampus as a cognitive map*. Clarendon Press. ISBN: 0-19-857206-9. URL: https://discovery.ucl.ac.uk/id/eprint/10103569/.

O'Keefe, John and Michael L. Recce (1993). "Phase relationship between hippocampal place units and the EEG theta rhythm". In: *Hippocampus* 3.3, pp. 317–330. ISSN: 1098-1063. DOI: 10.1002/hipo.450030307.

Pandarinath, Chethan, Daniel J. O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D. Stavisky, Jonathan C. Kao, Eric M. Trautmann, Matthew T. Kaufman, Stephen I. Ryu, Leigh R. Hochberg, J. A. Henderson, Krishna V. Shenoy, L. F. Abbott, and David Sussillo (2018). "Inferring single-trial neural population dynamics using sequential auto-encoders". In: *Nature methods*. DOI: 10.1038/s41592-018-0109-9.

Paninski, Liam, Jonathan Pillow, and Jeremy Lewi (2007). "Statistical models for neural encoding, decoding, and optimal stimulus design". In: *Computational Neuroscience: Theoretical Insights into Brain Function*. Elsevier, pp. 493–507. ISBN: 9780444528230. DOI: 10.1016/s0079-6123(06)65031-0.

Parisi, German I., Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter (2018). "Continual Lifelong Learning with Neural Networks: A Review". In: DOI: 10.48550/ARXIV.1802.07569.

Park, EunHye, Dino Dvorak, and André A. Fenton (2011). "Ensemble place codes in hippocampus: CA1, CA3, and dentate gyrus place cells have multiple place fields in large environments". In: *PLoS one*. DOI: 10.1371/journal.pone.0022349.

Park, Mijung, Gergo Bohner, and Jakob H. Macke (2015). "Unlocking neural population non-stationarities using hierarchical dynamics models". In: *Advances in Neural Information Processing Systems*. URL: https://papers.nips.cc/paper_files/paper/2015/hash/28dd2c7955ce926456240b2ff0100bde-Abstract.html.

Pastalkova, Eva, Vladimir Itskov, Asohan Amarasingham, and György Buzsáki (Sept. 2008). "Internally Generated Cell Assembly Sequences in the Rat Hippocampus". In: *Science* 321.5894, pp. 1322–1327. ISSN: 1095-9203. DOI: 10.1126/science.1159775.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). "PyTorch: An imperative style, high-performance deep learning library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 8024–8035. DOI: 10.48550/arXiv.1912.01703.

Patel, Jagdish, Shigeyoshi Fujisawa, Antal Berényi, Sébastien Royer, and György Buzsáki (2012). "Traveling theta waves along the entire septotemporal axis of the hippocampus". In: *Neuron* 75.3, pp. 410–417. DOI: 10.1016/j.neuron.2012.07.015.

Payeur, Alexandre, Jordan Guerguiev, Friedemann Zenke, Blake A. Richards, and Richard Naud (May 2021). "Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits". In: *Nature Neuroscience* 24.7, pp. 1010–1019. DOI: 10.1038/s41593-021-00857-x.

Pearson, Karl (1901). "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2.11, pp. 559–572. DOI: 10.1080/14786440109462720.

Pereira, Talmo D., Nathaniel Tabris, Arie Matsliah, David M. Turner, Junyu Li, Shruthi Ravindranath, Eleni S. Papadoyannis, Edna Normand, David S. Deutsch, Z. Yan Wang, Grace C. McKenzie-Smith, Catalin C. Mitelut, Marielisa Diez Castro, John D'Uva, Mikhail Kislin, Dan H. Sanes, Sarah D. Kocher, Samuel S.-H. Wang, Annegret L. Falkner, Joshua W. Shaevitz, and Mala Murthy (Apr. 2022). "SLEAP: A deep learning system for multi-animal pose tracking". In: *Nature Methods* 19.4, pp. 486–495. ISSN: 1548-7105. DOI: 10.1038/s41592-022-01426-1.

Petersen, Peter Christian and György Buzsáki (2020). "Cooling of medial septum reveals theta phase lag coordination of hippocampal cell assemblies". In: *Neuron* 107.4, pp. 731–744. DOI: 10.1016/j.neuron.2020.05.023.

Pfeiffer, Brad E. and David J. Foster (Apr. 2013). "Hippocampal place-cell sequences depict future paths to remembered goals". In: *Nature* 497.7447, pp. 74–79. ISSN: 1476-4687. DOI: 10.1038/nature12112.

Pillow, Jonathan W., Jonathon Shlens, Liam Paninski, Alexander Sher, Alan M. Litke, E. J. Chichilnisky, and Eero P. Simoncelli (July 2008). "Spatio-temporal correlations and visual signalling in a complete neuronal population". In: *Nature* 454.7207, pp. 995–999. ISSN: 1476-4687. DOI: 10.1038/nature07140.

Piray, Payam and Nathaniel D. Daw (Dec. 2021). "Linear reinforcement learning in planning, grid fields, and cognitive control". In: *Nature Communications* 12.1, p. 4942. ISSN: 2041-1723. DOI: 10.1038/s41467-021-25123-3.

Platt, Michael L. and Paul W. Glimcher (1999). "Neural correlates of decision variables in parietal cortex". In: *Nature*. DOI: 10.1038/22268.

Ramesh, Aditya, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever (2021). "Zero-shot text-to-image generation". In: *CoRR* abs/2102.12092. arXiv: 2102.12092. URL: https://arxiv.org/abs/2102.12092.

Rao, Rajesh P. N. and Dana H. Ballard (Jan. 1999). "Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects". In: *Nature Neuroscience* 2.1, pp. 79–87. DOI: 10.1038/4580.

Rashidy-Pour, A., F. Motamedi, and Z. Motahed-Larijani (1996). "Effects of reversible inactivations of the medial septal area on reference and working memory versions of the Morris water maze". In: *Brain research* 709.1, pp. 131–140. DOI: 10.1016/0006-8993(95)01323-7.

Rauch, Herbert E., F. Tung, and Charlotte T. Striebel (1965). "Maximum likelihood estimates of linear dynamic systems". In: *AIAA journal*. DOI: 10.2514/3.3166.

Raudies, Florian and Michael E. Hasselmo (June 2012). "Modeling boundary vector cell firing given optic flow as a cue". In: *PLOS Computational Biology* 8.6, e1002553. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1002553.

Redish, A. David and David S. Touretzky (1998). "The role of the hippocampus in solving the Morris water maze". In: *Neural computation* 10.1, pp. 73–111. DOI: 10.1162/089976698300017908.

Reifenstein, Eric Torsten, Ikhwan Bin Khalid, and Richard Kempter (2021). "Synaptic learning rules for sequence learning". In: *eLife* 10, e67171. DOI: 10.7554/elife.67171.sa2.

Richards, Blake A. and Timothy P. Lillicrap (Feb. 2019). "Dendritic solutions to the credit assignment problem". In: *Current Opinion in Neurobiology* 54, pp. 28–36. DOI: 10.1016/j.conb.2018.08.003.

Rolls, Edmund T. and Raymond P. Kesner (May 2006). "A computational theory of hippocampal function, and empirical tests of the theory". In: *Progress in Neurobiology* 79.1, pp. 1–48. ISSN: 0301-0082. DOI: 10.1016/j.pneurobio.2006.04.005.

Rübel, Oliver, Andrew Tritt, Ryan Ly, Benjamin K Dichter, Satrajit Ghosh, Lawrence Niu, Pamela Baker, Ivan Soltesz, Lydia Ng, Karel Svoboda, Loren Frank, and Kristofer E Bouchard (Oct. 2022). "The Neurodata Without Borders ecosystem for neurophysiological data science". In: *eLife* 11. ISSN: 2050-084X. DOI: 10.7554/elife.78362.

Russek, Evan M., Ida Momennejad, Matthew M. Botvinick, Samuel J. Gershman, and Nathaniel D. Daw (2017). "Predictive representations can link model-based

reinforcement learning to model-free mechanisms". In: *PLOS Computational Biology* 13.9. ISSN: 15537358. DOI: 10.1371/journal.pcbi.1005768.

Sacramento, João, Rui Ponte Costa, Yoshua Bengio, and Walter Senn (2018). "Dendritic cortical microcircuits approximate the backpropagation algorithm". In: *Advances in Neural Information Processing Systems*, pp. 8721–8732. DOI: 10.48550/arXiv.1810.11393.

Samsonovich, Alexei and Bruce L. McNaughton (1997). "Path integration and cognitive mapping in a continuous attractor neural network model". In: *The Journal of Neuroscience* 17.15, pp. 5900–5920. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.17-15-05900.1997.

Sanders, Honi, César Rennó-Costa, Marco Idiart, and John E. Lisman (2015). "Grid cells and place cells: An integrated view of their navigational and memory function". In: *Trends in Neurosciences* 38.12, pp. 763–775. ISSN: 0166-2236. DOI: 10.1016/j.tins.2015.10.004.

Sanders, Honi, Matthew A. Wilson, and Samuel J. Gershman (June 2020). "Hippocampal remapping as hidden state inference". In: *eLife* 9. DOI: 10.7554/elife.51140.

Sani, Omid G., Hamidreza Abbaspourazad, Yan T. Wong, Bijan Pesaran, and Maryam M. Shanechi (2021). "Modeling behaviorally relevant neural dynamics enabled by preferential subspace identification". In: *Nature Neuroscience* 24.1, pp. 140–149. DOI: 10.1038/s41593-020-00733-0.

Sargolini, Francesca, Marianne Fyhn, Torkel Hafting, Bruce L. McNaughton, Menno P. Witter, May-Britt Moser, and Edvard I. Moser (May 2006). "Conjunctive representation of position, direction, and velocity in entorhinal cortex". In: *Science* 312.5774, pp. 758–762. DOI: 10.1126/science.1125572.

Satoh, Y., S. Endo, T. Nakata, Y. Kobayashi, K. Yamada, T. Ikeda, A. Takeuchi, T. Hiramoto, Y. Watanabe, and T. Kazama (Aug. 2011). "ERK2 contributes to the control of social behaviors in mice". In: *The Journal of Neuroscience* 31.33, pp. 11953–11967. DOI: 10.1523/jneurosci.2349-11.2011.

Schaeffer, Rylan, Mikail Khona, and Ila Rani Fiete (2022). "No free lunch from deep learning in neuroscience: A case study through models of the entorhinal-

hippocampal circuit". In: *ICML 2022 2nd AI for Science Workshop*. URL: https://openreview.net/forum?id=mxi1xKzNFrb.

Schneegans, Susan, Tiffany Straza, Jake Lewis, Peter Gluckman, Binyam Mendisu, Eric D'Ortenzio, Evelyne Marche, Oriane Puéchal, Immaculada Perez, Yazdan Yazdanpanah, Ana Persic, Ferndanda Beigel, Simon Hodson, Peggy Oti-Boateng, Shamila Nair-Bedouelle, Peter McGrath, Edward Lempinen, April Tash, Alessandro Bello, and Ranasinghe Amaradasa (June 2021). *UNESCO Science Report: the Race Against Time for Smarter Development (2021)*. ISBN: 978-92-3-100450-6. DOI: 10.18356/9789210058575c010.

Schneider, Steffen, Jin Hwa Lee, and Mackenzie Weygandt Mathis (2023). "Learnable latent embeddings for joint behavioural and neural analysis". In: *Nature*. DOI: 10.1038/s41586-023-06031-6.

Schultz, Wolfram, Peter Dayan, and P. Read Montague (1997). "A neural substrate of prediction and reward". In: *Science* 275.5306, p. 6. DOI: 10.1126/science.275.5306.1593.

Scoville, W. B. and B. Milner (Feb. 1957). "Loss of recent memory after bilateral hippocampal lesions". In: *Journal of Neurology, Neurosurgery, and Psychiatry* 20.1, pp. 11–21. ISSN: 0022-3050. DOI: 10.1136/jnnp.20.1.11.

Seymour, Ben, John P. O'Doherty, Peter Dayan, Martin Koltzenburg, Anthony K. Jones, Raymond J. Dolan, Karl J. Friston, and Richard S. Frackowiak (2004). "Temporal difference models describe higher-order learning in humans". In: *Nature* 429.6992, pp. 664–667. DOI: 10.1038/nature02581.

Sharma, Sugandha, Sarthak Chandra, and Ila Rani Fiete (July 2022). "Content addressable memory without catastrophic forgetting by heteroassociation with a fixed scaffold". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. PMLR, pp. 19658–19682. DOI: 10.48550/arXiv.2202.00159.

Skaggs, William E., J. J. Knierim, H. S. Kudrimoti, and Bruce L. McNaughton (1995). "A model of the neural basis of the rat's sense of direction". In: *Advances in neural information processing systems* 7, pp. 173–180. URL: https://proceedings.neurips.cc/paper/1994/file/024d7f84fff11dd7e8d9c510137a2381-Paper.pdf.

Skaggs, William E. and Bruce L. McNaughton (Mar. 1996a). "Replay of neuronal firing sequences in rat hippocampus during sleep following spatial experience". In: *Science* 271.5257, pp. 1870–1873. DOI: 10.1126/science.271.5257.1870.

Skaggs, William E., Bruce L. McNaughton, Matthew A. Wilson, and Carol A. Barnes (1996b). "Theta phase precession in hippocampal neuronal populations and the compression of temporal sequences". In: *Hippocampus* 6.2, pp. 149–172. DOI: 10.1002/(sici)1098-1063(1996)6:2<149::aid-hipo6>3.0.co;2-k.

Smith, Anne C. and Emery N. Brown (2003). "Estimating a state-space model from point process observations". In: *Neural Computation*. DOI: 10.1162/089976603765202622.

Sohl-Dickstein, Jascha, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli (July 2015). "Deep unsupervised learning using nonequilibrium thermodynamics". In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. PMLR, pp. 2256–2265. DOI: 10.48550/arXiv.1503.03585.

Soldado-Magraner, Saray, Joana Soldado-Magraner, Gabriela Michel, Emmanuel Marquez-Legorreta, and Artemis Koumoundourou (2023). *TReND-CaMinA: Computational Neuroscience and Machine Learning in Africa*. https://trendinafrica.org/trend-camina/. Accessed: 2025-06-23.

Solstad, Trygve, Charlotte N. Boccara, Emilio Kropff, May-Britt Moser, and Edvard I. Moser (2008). "Representation of geometric borders in the entorhinal cortex". In: *Science* 322.5909, pp. 1865–1868. ISSN: 0036-8075. DOI: 10.1126/science.1166466.

Sorscher, Ben, Gabriel C. Mel, Samuel A. Ocko, Lisa M. Giocomo, and Surya Ganguli (2023). "A unified theory for the computational and mechanistic origins of grid cells". In: *Neuron* 111.1, 121–137.e13. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2022.10.003.

Spiers, Hugo J., Robin Hayman, Aleksandar Jovalekic, Elizabeth Marozzi, and Kathryn J. Jeffery (2015). "Place field repetition and purely local remapping in a multicompartment environment". In: *Cerebral Cortex* 25.1. ISBN: 1047-3211, pp. 10–25. ISSN: 14602199. DOI: 10.1093/cercor/bht198.

Spiers, Hugo J. and Eleanor A. Maguire (July 2006). “Thoughts, behaviour, and brain dynamics during navigation in the real world”. In: *NeuroImage* 31.4, pp. 1826–1840. DOI: 10.1016/j.neuroimage.2006.01.037.

Squire, Larry R. (July 1992). “"Memory and the hippocampus: A synthesis from findings with rats, monkeys, and humans": Correction”. In: *Psychological Review* 99.3, pp. 582–582. DOI: 10.1037/0033-295x.99.3.582.

Squire, Larry R., Anna S. van der Horst, Susan G. R. McDuff, Jennifer C. Frascino, Ramona O. Hopkins, and Kristin N. Mauldin (2010). “Role of the hippocampus in remembering the past and imagining the future”. In: *Proceedings of the National Academy of Sciences*. DOI: 10.1073/pnas.1014391107.

Stachenfeld, Kimberly L., Matthew M. Botvinick, and Samuel J. Gershman (2014). “Design principles of the hippocampal cognitive map”. In: *Advances in neural information processing systems* 27. URL: https://papers.nips.cc/paper_files/paper/2014/file/6083b607d0b81940c0280e465c79f5d5-Paper.pdf.

Stachenfeld, Kimberly L., Matthew M. Botvinick, and Samuel J. Gershman (Nov. 2017). “The hippocampus as a predictive map”. In: *Nature Neuroscience* 20.11, pp. 1643–1653. ISSN: 1097-6256, 1546-1726. DOI: 10.1038/nn.4650.

Stimberg, Marcel, Romain Brette, and Dan F. M. Goodman (Aug. 2019). “Brian 2, an intuitive and efficient neural simulator”. In: *eLife* 8, e47314. ISSN: 2050-084X. DOI: 10.7554/eLife.47314.

Strange, Bryan A., Menno P. Witter, Ed S. Lein, and Edvard I. Moser (Sept. 2014). “Functional organization of the hippocampal longitudinal axis”. In: *Nature Reviews Neuroscience* 15.10, pp. 655–669. DOI: 10.1038/nrn3785.

Sutton, Richard S. (Aug. 1988). “Learning to predict by the methods of temporal differences”. In: *Machine Learning* 3.1, pp. 9–44. DOI: 10.1007/bf00115009.

Sutton, Richard S. and Andrew G. Barto (1998). *Introduction to reinforcement learning*. MIT Press. ISBN: 0262193981. URL: http://incompleteideas.net/book/the-book-2nd.html.

Tanni, Sander, William de Cothi, and Caswell Barry (June 2021). “State transitions in the statistically stable place cell population are determined by rate of perceptual change”. In: *bioRxiv*. DOI: 10.1101/2021.06.16.448638.

Tanni, Sander, William de Cothi, and Caswell Barry (2022). "State transitions in the statistically stable place cell population correspond to rate of perceptual change". In: *Current Biology*. ISSN: 0960-9822. DOI: 10.1016/j.cub.2022.06.046.

Taube, J. S., Robert U. Muller, and J. B. Ranck (1990). "Head-direction cells recorded from the postsubiculum in freely moving rats. I. Description and quantitative analysis". In: *The Journal of Neuroscience* 10.2, pp. 420–435. ISSN: 0270-6474. DOI: 10.1523/jneurosci.10-02-00420.1990.

Tipping, Michael E. and Christopher M. Bishop (1999). "Probabilistic principal component analysis". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 61.3, pp. 611–622. URL: https://www.cs.columbia.edu/~blei/seminar/2020-representation/readings/TippingBishop1999.pdf.

Todorov, Emanuel (July 2009). "Efficient computation of optimal actions". In: *Proceedings of the National Academy of Sciences* 106.28, pp. 11478–11483. DOI: 10.1073/pnas.0710743106.

Todorov, Emanuel, Tom Erez, and Yuval Tassa (Oct. 2012). "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. DOI: 10.1109/iros.2012.6386109.

Tolman, Edward C. (1948). "Cognitive maps in rats and men". In: *Psychological review*. DOI: 10.1037/h0061626.

Tolman, Edward C. and C. H. Honzik (1930). "Introduction and removal of reward, and maze performance in rats". In: *University of California Publications in Psychology* 4, pp. 257–275.

Treves, Alessandro and Edmund T. Rolls (June 1994). "Computational analysis of the role of the hippocampus in memory". In: *Hippocampus* 4.3, pp. 374–391. ISSN: 1098-1063. DOI: 10.1002/hipo.450040319.

Urbanczik, Robert and Walter Senn (Feb. 2014). "Learning by the dendritic prediction of somatic spiking". In: *Neuron* 81.3, pp. 521–528. ISSN: 08966273. DOI: 10.1016/j.neuron.2013.11.030.

Uria, Benigno, Borja Ibarz, Andrea Banino, Vinicius Zambaldi, Dharshan Kumaran, Demis Hassabis, Caswell Barry, and Charles Blundell (Nov. 2020). "A model of

egocentric to allocentric understanding in mammalian brains". In: DOI: 10.110 1/2020.11.11.378141.

Vaart, Aad W. van der (2000). *Asymptotic statistics*. Vol. 3. Cambridge university press.

Vafidis, Pantelis, David Owald, Tiziano D'Albis, and Richard Kempter (June 2022). "Learning accurate path integration in ring attractor models of the head direction system". In: *eLife* 11, e69841. ISSN: 2050-084X. DOI: 10.7554/eLife.69841.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need". In: *CoRR* abs/1706.03762. DOI: 10.48550/arXiv.1706.03762.

Vértes, Eszter and Maneesh Sahani (2019). "A neurally plausible model learns successor representations in partially observable environments". In: *Advances in Neural Information Processing Systems* 32. DOI: 10.48550/arXiv.1906.09480.

Vries, Saskia EJ de, Joshua H Siegle, and Christof Koch (July 2023). "Sharing neurophysiology data from the Allen Brain Observatory". In: *eLife* 12. ISSN: 2050-084X. DOI: 10.7554/elife.85550.

Wang, Jack, Aaron Hertzmann, and David J. Fleet (2005). "Gaussian process dynamical models". In: *Advances in neural information processing systems* 18. URL: https://papers.nips.cc/paper_files/paper/2005/hash/ccd450 07df44dd0f12098f486e7e8a0f-Abstract.html.

Whittington, James C. R., Timothy H. Muller, Shirley Mark, Guifen Chen, Caswell Barry, Neil Burgess, and Timothy E. J. Behrens (Nov. 2020). "The Tolman-Eichenbaum Machine: Unifying space and relational memory through generalization in the hippocampal formation". In: *Cell* 183.5, 1249–1263.e23. ISSN: 00928674. DOI: 10.1016/j.cell.2020.10.024.

Whittington, James C.R., William Dorrell, Timothy E.J. Behrens, Surya Ganguli, and Mohamady El-Gaby (Jan. 2025). "A tale of two algorithms: Structured slots explain prefrontal sequence memory and are unified with hippocampal cognitive maps". In: *Neuron* 113.2, 321–333.e6. ISSN: 0896-6273. DOI: 10.1016 /j.neuron.2024.10.017.

Widloski, John and Ila Rani Fiete (2014). "A model of grid cell development through spatial exploration and spike time-dependent plasticity". In: *Neuron* 83.2, pp. 481–495. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2014.06.018.

Wikenheiser, Andrew M. and A. David Redish (Jan. 2015). "Hippocampal theta sequences reflect current goals". In: *Nature Neuroscience* 18.2, pp. 289–294. DOI: 10.1038/nn.3909.

Williams, Alex, Anthony Degleris, Yixin Wang, and Scott W. Linderman (2020). "Point process models for sequence detection in high-dimensional neural spike trains". In: *Advances in Neural Information Processing Systems*. DOI: 10.48550/arXiv.2010.04875.

Wills, Tom J., Colin Lever, Francesca Cacucci, Neil Burgess, and John O'Keefe (May 2005). "Attractor Dynamics in the Hippocampal Representation of the Local Environment". In: *Science* 308.5723, pp. 873–876. ISSN: 1095-9203. DOI: 10.1126/science.1108905.

Wilson, Greg, Jennifer Bryan, Karen Cranston, Justin Kitzes, Lex Nederbragt, and Tracy K. Teal (June 2017). "Good enough practices in scientific computing". In: *PLOS Computational Biology* 13.6. Ed. by Francis Ouellette, e1005510. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1005510.

Wilson, Matthew A. and Bruce L. McNaughton (1993). "Dynamics of the hippocampal ensemble code for space". In: *Science*. DOI: 10.1126/science.8351520.

Wilson, Matthew A. and Bruce L. McNaughton (July 1994). "Reactivation of hippocampal ensemble memories during sleep". In: *Science* 265.5172, pp. 676–679. DOI: 10.1126/science.8036517.

Wilt, Brian A., Laurie D. Burns, Eric Tatt Wei Ho, Kunal K. Ghosh, Eran A. Mukamel, and Mark J. Schnitzer (2009). "Advances in light microscopy for neuroscience". In: *Annual review of neuroscience*. DOI: 10.1146/annurev.neuro.051508.135540.

Wu, Anqi, Nicholas A. Roy, Stephen Keeley, and Jonathan W. Pillow (2017). "Gaussian process based nonlinear latent structure discovery in multivariate spike train data". In: *Advances in Neural Information Processing Systems*. URL: https://papers.nips.cc/paper_files/paper/2017/hash/b3b4d2dbedc99fe843fd3dedb02f086f-Abstract.html.

Yamaguchi, Yoko, Yoshito Aota, Bruce L. McNaughton, and Peter Lipa (June 2002). "Bimodality of theta phase precession in hippocampal place cells in freely running rats". In: *Journal of Neurophysiology* 87.6, pp. 2629–2642. DOI: 10.1152/jn.2002.87.6.2629.

Yang, Yibo, Stephan Mandt, and Lucas Theis (2022). *An Introduction to Neural Data Compression*. DOI: 10.48550/ARXIV.2202.06533.

Yu, Byron M., John P. Cunningham, Gopal Santhanam, Stephen Ryu, Krishna V. Shenoy, and Maneesh Sahani (2008a). "Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity". In: *Advances in Neural Information Processing Systems*.

Yu, Byron M., John P. Cunningham, Krishna V. Shenoy, and Maneesh Sahani (2008b). "Neural decoding of movements: From linear to nonlinear trajectory models". In: *Neural Information Processing: 14th International Conference, ICONIP 2007, Kitakyushu, Japan, November 13-16, 2007, Revised Selected Papers, Part I 14*. Springer.

Yu, Byron M., Krishna V. Shenoy, and Maneesh Sahani (2006). "Expectation propagation for inference in non-linear dynamical models with Poisson observations". In: *2006 IEEE Nonlinear Statistical Signal Processing Workshop*. IEEE. DOI: 10.1109/nsspw.2006.4378825.

Zador, Anthony, Sean Escola, Blake Richards, Bence Ölveczky, Yoshua Bengio, Kwabena Boahen, Matthew Botvinick, Dmitri Chklovskii, Anne Churchland, Claudia Clopath, James DiCarlo, Surya Ganguli, Jeff Hawkins, Konrad Körding, Alexei Koulakov, Yann LeCun, Timothy Lillicrap, Adam Marblestone, Bruno Olshausen, Alexandre Pouget, Cristina Savin, Terrence Sejnowski, Eero Simoncelli, Sara Solla, David Sussillo, Andreas S. Tolias, and Doris Tsao (Mar. 2023). "Catalyzing next-generation Artificial Intelligence through NeuroAI". In: *Nature Communications* 14.1. ISSN: 2041-1723. DOI: 10.1038/s41467-023-37180-x.

Zentall, Thomas R. (Mar. 2015). "When animals misbehave: Analogs of human biases and suboptimal choice". In: *Behavioural Processes* 112, pp. 3–13. DOI: 10.1016/j.beproc.2014.08.001.

Zhang, K. (Mar. 1996). "Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: A theory". In: *The Journal of Neuroscience* 16.6, pp. 2112–2126. DOI: 10.1523/jneurosci.16-06-02112.1996.

Zhao, Yuan and Il Memming Park (2017). "Variational latent gaussian process for recovering single-trial dynamics from population spike trains". In: *Neural Computation*. DOI: 10.1162/neco_a_00953.

Zhou, Ding and Xue-Xin Wei (2020). "Learning identifiable and interpretable latent models of high-dimensional neural activity using pi-VAE". In: *Advances in Neural Information Processing Systems*. URL: https://papers.nips.cc/paper/2020/hash/510f2318f324cf07fce24c3a4b89c771-Abstract.html.

Znamenskiy, Petr and Anthony M. Zador (2013). "Corticostriatal neurons in auditory cortex drive decisions during auditory discrimination". In: *Nature*. DOI: 10.1038/nature12077.

Zoltowski, David, Jonathan Pillow, and Scott W. Linderman (2020). "A general recurrent state space framework for modeling neural dynamics during decision-making". In: *International Conference on Machine Learning*. PMLR, pp. 11680–11691. URL: https://proceedings.mlr.press/v119/zoltowski20a.html.

Appendix A

Appendix to Chapter 1

A.1 Code Availability

Code used to generate the results in Chapter 1, as well as the RatInABox package itself, is available at `https://github.com/RatInABox-Lab/RatInABox`.

A.2 Model and Feature Specifications

The following section describes in mathematical detail the models used within RatInABox.

A.2.1 Motion Model Details

Temporally continuous random motion

The random motion model is based on the Ornstein Uhlenbeck (OU) process, $\mathsf{X}_{\theta,\lambda,\mu}(t)$, a stochastic process satisfying the Langevin differential equation

$$\begin{aligned} \mathsf{X}_{\theta,\lambda,\mu}(t+dt) &= \mathsf{X}_{\theta,\lambda,\mu}(t) + d\mathsf{X}_{\theta,\lambda,\mu}(t), \\ d\mathsf{X}_{\theta,\lambda,\mu}(t) &= \theta(\mu - \mathsf{X}_{\theta,\lambda,\mu}(t))dt + \lambda\eta(t)\sqrt{dt} \end{aligned} \tag{A.1}$$

where $\eta(t) \sim \mathcal{N}(0,1)$ is Gaussian white noise and θ, λ and μ are constants. The first term in the update equation drives decay of $\mathsf{X}_{\theta,\lambda,\mu}(t)$ towards the mean μ. The second term is a stochastic forcing term, driving randomness. These stochastic processes are well studied; their unconditioned covariance

across time is

$$\langle \mathsf{X}_{\theta,\lambda,\mu}(t)\mathsf{X}_{\theta,\lambda,\mu}(t')\rangle = \frac{\lambda^2}{2\theta}e^{-\theta|t-t'|}. \tag{A.2}$$

Thus $\mathsf{X}_{\theta,\lambda,\mu}(t)$ decorrelates smoothly over a timescale of $\tau = 1/\theta$. Over long periods $\mathsf{X}_{\theta,\lambda,\mu}(t)$ is stochastic and therefore unpredictable. Its long-run stationary probability distribution is a Gaussian with mean μ and standard deviation $\sigma = \sqrt{\lambda^2/2\theta}$. The Ornstein Uhlenbeck process can be re-parameterised in terms of these more intuitive parameters (the decorrelation timescale τ and the long-run standard deviation σ) using the transformations

$$\theta = \frac{1}{\tau} \quad , \quad \lambda = \sqrt{\frac{2\sigma^2}{\tau}}, \tag{A.3}$$

to give

$$\begin{aligned} \mathsf{X}_{\tau,\sigma,\mu}(t+dt) &= \mathsf{X}_{\tau,\sigma,\mu}(t) + d\mathsf{X}_{\tau,\sigma,\mu}(t), \\ d\mathsf{X}_{\tau,\sigma,\mu}(t) &= \frac{1}{\tau}(\mu - \mathsf{X}_{\tau,\sigma,\mu}(t))dt + \sqrt{\frac{2\sigma^2}{\tau}}\eta(t)\sqrt{dt}. \end{aligned} \tag{A.4}$$

Ornstein Uhlenbeck processes have the appealing property that they are temporally continuous (their statistics are independent of dt) and allow for easy control of the long-run standard deviation and the decorrelation timescale of the stochastic variable. For these reasons they are used to model rotational and linear velocities within RatInABox.

2D motion For 2D locomotion, the `Agent`'s rotational velocity $\omega(t) = \dot{\theta}_{\mathsf{v}}(t)$ and linear speed, $\mathsf{v}_{2D}(t) = \|\mathbf{v}(t)\|$, are sampled from independent OU processes. This is because, as shown in section 1.3, they have decorrelation timescales differing by an order of magnitude. Rotational velocity is sampled from a standard Ornstein Uhlenbeck process with zero mean. Linear speed is also sampled from an Ornstein Uhlenbeck process with one additional transform applied in order to match the observation that linear speeds have a Rayleigh,

not normal, distribution.

$$\omega(t) \sim \mathsf{X}_{\tau_\omega,\sigma_\omega,0}(t), \tag{A.5}$$

$$\mathsf{v}_{2D}(t) = R_{\sigma_\mathsf{v}}(z(t)) \quad \text{where} \quad z(t) \sim \mathsf{X}_{\tau_\mathsf{v},1,0}(t), \tag{A.6}$$

where $R_\sigma(x)$ is a monotonic transformation which maps a normally distributed random variable $x \sim \mathcal{N}(0,1)$ to one with a Rayleigh distribution of scale parameter σ. σ corresponds to the mode, or ≈ 0.8 times the mean, of the Rayleigh distribution.

$$R_\sigma(x) = \sigma\sqrt{-2\ln\Big(1-\frac{1}{2}\Big[1+\mathrm{erf}\Big(\frac{x}{\sqrt{2}}\Big)\Big]\Big)}. \tag{A.7}$$

The parameters $\{\tau_\omega, \sigma_\omega, \tau_\mathsf{v}, \sigma_\mathsf{v}\}$ are fitted from real open field 2D locomotion data in Figure 1.2 or can be set by the user.

Full trajectories are then sampled as follows: First the rotational and linear velocities are updated according to eqs. (A.5) and (A.6) (and additional considerations for walls, see next section). Next the velocity direction, $\theta_\mathsf{v}(t)$ – defined as the angle of the velocity vector measured anticlockwise from the x-direction – is updated according to the rotational velocity, $\omega(t)$.

$$\theta_\mathsf{v}(t) = \Big(\theta_\mathsf{v}(t-dt) + \omega(t)dt\Big) \mod 2\pi. \tag{A.8}$$

This is combined with the linear speed, $\mathsf{v}_{2D}(t)$ to calculate new total velocity vector, $\mathbf{v}(t)$.

$$\mathbf{v}(t) = \mathsf{v}_{2D}(t)\begin{bmatrix}\cos\theta_\mathsf{v}(t)\\ \sin\theta_\mathsf{v}(t)\end{bmatrix}. \tag{A.9}$$

Finally position, $\mathbf{x}(t)$, is updated by integrating along the total velocity vector to give a continuous and smooth, but over long time periods random, motion

trajectory.

$$\mathbf{x}(t) = \mathbf{x}(t - dt) + \mathbf{v}(t)dt. \tag{A.10}$$

1D motion Motion in 1D is more simple than motion in 2D. Velocity is also modelled as an Ornstein Uhlenbeck process without the Rayleigh transform. In this case a non-zero mean, μ_{v}, corresponding to a directional bias in the motion, can be provided by the user. In summary:

$$\mathsf{v}_{1D}(t) \sim \mathsf{X}_{\tau_{\mathsf{v}},\sigma_{\mathsf{v}},\mu_{\mathsf{v}}}(t), \tag{A.11}$$

$$x(t) = x(t - dt) + \mathsf{v}_{1D}(t)dt. \tag{A.12}$$

External velocity control

It is possible to provide an external velocity signal controlling the `Agent`'s motion. After the random motion update (as described above) is applied, if an external velocity $\mathbf{v}_{\text{drift}}(t)$ is provided by the user, an additional update to the velocity vector is performed

$$d\mathbf{v}(t + dt) = \frac{1}{\tau_{\text{drift}}}(\mathbf{v}_{\text{drift}}(t) - \mathbf{v}(t))dt. \tag{A.13}$$

In cases where $\tau_{\text{drift}} >> \tau_{\mathsf{v}}$ the net update to the velocity (random update and drift update) is dominated by the random component. When $\tau_{\text{drift}} << \tau_{\mathsf{v}}$ the update is dominated by the drift component. I define $\tau_{\text{drift}} := \tau_{\mathsf{v}}/k$ where k is an argument also provided by the user. To good approximation for large $k >> 1$ the `Agent` velocity closely tracks the drift velocity at all times and is not random whilst for $k << 1$ the drift velocity is ignored and the motion is entirely random.

Motion near walls in 2D

An important feature is the ability to generate `Environments` with arbitrary arrangements of walls (aka 'barriers' or 'boundaries'). Walls are meaningful

only if they appropriately constrain the motion of the `Agent`. For biological agents this means three things:

1. The `Agent` cannot travel through a wall.
2. The `Agent` slows down upon approaching a wall to avoid a full-speed collision.
3. There may be a bias called "thigmotaxis" for the `Agent` to stay near walls.

The motion model replicates these three effects as follows:

Collision detection To avoid travelling through walls, if a collision is detected the velocity is elastically reflected off the wall (normal component is flipped). The speed is then scaled to one half the average motion speed, $\mathtt{v}_{2D}(t) = 0.5\sigma_{\mathtt{v}}$.

Wall repulsion Spring-deceleration model In order to slow down *before* colliding with a wall the `Agent` feels an acceleration, perpendicular to the wall, whenever it is within a small distance, d_{wall}, of the wall.

$$\dot{\mathbf{v}}(t) = k_1 \sum_{\text{walls},j} \mathbf{n}_j \begin{cases} \frac{(s\cdot\sigma_{\mathtt{v}})^2}{d_{\text{wall}}^2} \cdot (d_{\text{wall}} - d_{\perp,j}(t)) & \text{if } d_{\perp,j}(t) \leq d_{\text{wall}}, \\ 0 & \text{if } d_{\perp,j}(t) > d_{\text{wall}}. \end{cases} \tag{A.14}$$

$d_{\perp,j}(t)$ is the perpendicular distance from the `Agent` to the j^{th} wall, $\mathbf{n}_j$ is the perpendicular norm of the j^{th} wall (the norm pointing towards the `Agent`) and k_1 & s are constants (explained later). d_{wall} is the distance from the wall at which the `Agent` starts to feel the deceleration, defaulting to $d_{\text{wall}} = 0.1$ m.

Note that this acceleration is identical to that of an oscillating spring-mass where the base of the spring is attached a distance d_{wall} from the wall on a perpendicular passing through the `Agent`. The spring constant is tuned such that a mass starting with initial velocity towards the wall of $-s\sigma_{\mathtt{v}}\mathbf{n}_j$ would stop *just* before the wall. In summary, for $k_1 = 1$, if the `Agent` approaches the wall head-on at speed of $s\sigma_{\mathtt{v}}$ (s times its mean speed) this deceleration will just be enough to avoid a collision.

s is the unitless wall repel strength parameter (default $s = 1$). When it is high, walls repel the agent strongly (only fast initial speeds will result in the agent reaching the wall) and when it is low, walls repel weakly (even very slow initial speeds will not be slowed done by the spring dynamics). When $s = 0$ wall repulsion is turned off entirely.

Conveyor-belt modelA second (similar, but not exactly equivalent) way to slow down motion near a wall is to consider a hypothetical conveyor belt near the wall. This conveyor belt has a non-uniform velocity pointing away from the wall of

$$\dot{\mathbf{x}}(t) = k_2 \sum_{\text{walls},j} \mathbf{n}_j \begin{cases} s \cdot \sigma_{\mathtt{v}} \left(1 - \sqrt{1 - \frac{(d_{\text{wall}} - d_{\perp,j}(t))^2}{d_{\text{wall}}^2}} \right) & \text{if } d_{\perp,j}(t) \leq d_{\text{wall}}, \\ 0 & \text{if } d_{\perp,j}(t) > d_{\text{wall}}. \end{cases} \tag{A.15}$$

When the `Agent` is close to the wall the hypothetical conveyor-belt moves it backwards on each time step, effectively slowing it down. Note that this velocity is identical to that of a spring-mass attached to the wall with initial velocity $s\sigma_{\mathtt{v}}\mathbf{n}_j$ away from the wall and spring constant tuned to stop the mass just before it reaches a distance d_{wall}. In summary, for $k_2 = 1$, if the `Agent` approaches the wall head-on at speed of $s\sigma_{\mathtt{v}}$ the conveyor belt will just be fast enough to bring it to a halt at the location of the wall.

Wall attraction (thigmotaxis) Although similar, there is an exploitable difference between the 'spring-deceleration' and 'conveyor-belt' models: the 'conveyor-belt' changes the `Agent`'s position, $\mathbf{x}(t)$, on each step but not its internal velocity variable $\mathbf{v}(t)$. As as result (and as the conveyor-belt intuition suggests) it will slow down the `Agent`'s approach towards the wall without causing it to turn around. This creates a "lingering" or "thigmotactic" effect whereby whenever the `Agent` heads towards a wall it may carry on doing so, without collision, for some time until the stochastic processes governing its motion (Appendix A.2.1) cause it to turn. Conversely the 'spring-deceleration' model has no "thigmotactic" effect since it actively changes the internal velocity

variable causing the `Agent` to turn around or "bounce" off the walls.

The relative strengths of these two effects, k_1 and k_2, are controlled by a single thigmotaxis parameter, $\lambda_{\text{thig}} \in [0, 1]$ which governs the trade-off between these two models.

$$k_1 = 3(1 - \lambda_{\text{thig}})^2, \qquad k_2 = 6\lambda_{\text{thig}}^2. \tag{A.16}$$

When $\lambda_{\text{thig}} = 1$ only the conveyor belt model is active giving a strong thigmotactic effects. When $\lambda_{\text{thig}} = 0$ only the spring-deceleration model is active giving no thigmotactic effect. By default $\lambda_{\text{thig}} = 0.5$. The constants 3 and 6 are tuning parameters chosen by hand in order that direct collisions with the walls are rare but not impossible.

Although this procedure, intended to smoothly slow the `Agent` near a wall, may seem complex, it has two advantages: Firstly, deceleration near walls is smooth, becoming stronger as the `Agent` gets nearer and so induces no physically implausible discontinuities in the velocity. Secondly, it provides a tunable way by which to control the amount of thigmotaxis (evidenced in Figure 1.2c,d). Recall that these equations only apply to motion very near the wall ($< d_{\text{wall}}$) and they can be turned off entirely ($s = 0$)).

Importing trajectories

Users can override the random motion model by importing their own trajectory with `Agent.import_trajectory(times,positions)` where `times` is an array of times (not necessarily evenly spaced) and `positions` is an array of positions at each time. The trajectory is then interpolated using `scipy.interpolate`'s `interp1d` function following which the standard RatInABox `Agent.update(dt)` API is called to move the `Agent` to a new position a time `dt` along the imported trajectory.

When moving along imported trajectories the `Agent` will not be subject to the wall repel nor wall collision effects described above.

Head direction

As well as position and velocity `Agents` have a head direction, $\hat{\mathbf{h}}(t)$. Head direction is used by various cell types to determine firing rate including `HeadDirectionCells` and (egocentric) `VectorCells`. By default, head direction is just the smoothed-then-normalised velocity vector, updated on each timestep as follows:

$$\mathbf{h}(t+dt) = \left(1 - \frac{dt}{\tau_h}\right)\hat{\mathbf{h}}(t) + \frac{dt}{\tau_h}\frac{\mathbf{v}(t)}{\|\mathbf{v}(t)\|} \tag{A.17}$$

$$\hat{\mathbf{h}}(t+dt) = \frac{\mathbf{h}(t+dt)}{\|\mathbf{h}(t+dt)\|}. \tag{A.18}$$

By default the amount of smoothing is very small (in 2D $\tau_h = 0.15$, in 1D there is no smoothing at all) meaning that, to a good approximation, head direction is simply the normalised velocity vector at time t, $\hat{\mathbf{h}}(t) \approx \hat{\mathbf{v}}(t)$. However by storing head direction as an independent variable, this makes available the possibility for users to craft their own, potentially more complex, head direction dynamics if desired.

I also define the head direction angle $\phi_h(t)$ aka. the angle of head direction vector measured clockwise from the x-axis.

A.2.2 Distance Measures

In many of the cell models it is necessary to calculate the "distance" between two locations in the `Environment` (for example to calculate the firing rate of a Gaussian `PlaceCell`). This might depend on the type of geometry being used and the arrangement of walls in the `Environment`. There are three types of geometry currently supported:

$$\texttt{euclidean: } d(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\| \tag{A.19}$$

$$\texttt{geodesic: } d(\mathbf{x}_1, \mathbf{x}_2) = \text{length of shortest wall-avoiding path between } \mathbf{x}_1 \text{ and } \mathbf{x}_2 \tag{A.20}$$

$$\texttt{line\_of\_sight: } d(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} \|\mathbf{x}_1 - \mathbf{x}_2\| & \text{if no obstructing wall} \\ \infty & \text{otherwise} \end{cases} \tag{A.21}$$

By default RatInABox typically uses `geodesic` distance, except in `Environment`s with more than one additional wall where calculating the shortest path becomes computationally expensive. In these cases `line_of_sight` distance is typically used instead. Furthermore, in `Environment`s with periodic boundary conditions these distance measures will respect the periodicity by always using the shortest path between two points, wrapping around boundaries if necessary. These geometry considerations are what allow RatInABox cell classes to interact sensibly with walls (e.g. by default place cells won't bleed through walls, as observed in the brain). This is hereon referred to as the "environmental-distance".

A.2.3 Cell Model Specifications

In the following section, mathematical models are listed for most of the default provided `Neurons` subclasses, including all those covered in this chapter. More cell types and documentation can be found on the codebase. Readers will note that, oftentimes, parameters are set randomly at the point of initialisation (e.g. where the place cells are located, the orientation of grid cells, the angular preference of boundary vector cells etc.). Many of these random parameters are all set as class attributes and so can be redefined after initialisation if necessary. For simplicity here I describe default behaviour only.

Maximum and minimum firing rates For most cell classes it is also possible to set their maximum and minimum firing rates ($f_{\max}$, $f_{\min}$). For simplicity

the formulae provided below are written such that they have a maximum firing rate of 1.0 Hz and minimum firing rate of 0.0 Hz but readers should be aware that after evaluation these firing rates are linearly scaled according to

$$F(t) \leftarrow (f_{\max} - f_{\min})F(t) + f_{\min}. \tag{A.22}$$

Noise By default all `Neurons` are noiseless with their firing rates entirely determined by the deterministic mathematical models given below. Smooth Ornstein Uhlenbeck sampled random noise of coherence timescale τ_η and magnitude σ_η can be added:

$$\eta(t) \sim \mathsf{X}_{\tau_\eta,\sigma_\eta,0}(t) \tag{A.23}$$

$$F(t) \leftarrow F(t) + \eta(t) \tag{A.24}$$

Rates vs. Spikes RatInABox `Neurons` are fundamentally rate-based. This means that their firing rate is a continuous function of time. Simultaneously, at every time-step, spikes are sampled from this firing rate and saved into the history dataframe in case spiking data is required:

$$\mathsf{P}(\texttt{Neuron}\ i \text{ spikes in } [t, t+dt]) = F_i(t)dt. \tag{A.25}$$

`PlaceCells`

A set of locations (the centre of the place fields), $\{\mathbf{x}_i^{\text{PC}}\}$, is randomly sampled from the `Environment`. By default these locations sit on a grid uniformly spanning the `Environment` to which a small amount of random jitter, half the scale of the sampled grid, is added. Thus place cell locations appear 'random' but initialising in this way ensures all parts of the `Environment` are approximately evenly covered with the same density of place fields.

The environmental-distance from the `Agent` to the place field centres is calculated ($d_i(t) = d(\mathbf{x}_i^{\text{PC}}, \mathbf{x}(t))$. The firing rate is then determined by one of

the following functions (defaulting to $F^{\texttt{gaussian}}$):

$$F_i^{\texttt{gaussian}}(t) = e^{-d_i^2/2w_i^2} \tag{A.26}$$

$$F_i^{\texttt{gaussian\_threshold}}(t) = \max\left(0, \frac{e^{-d_i^2/2w_i^2} - e^{-1/2}}{1 - e^{-1/2}}\right) \tag{A.27}$$

$$F_i^{\texttt{diff\_of\_gaussians}}(t; r = 1.5) = \frac{e^{-d_i^2/2w_i^2} - (1/r^2)e^{-d_i^2/2(rw_i)^2}}{1 - 1/r^2} \tag{A.28}$$

$$F_i^{\texttt{top\_hat}}(t) = \begin{cases} 1 & \text{if } d_i \leq w_i \\ 0 & \text{otherwise} \end{cases} \tag{A.29}$$

$$F_i^{\texttt{one\_hot}}(t) = \delta(i == \text{argmin}_j(d_j)). \tag{A.30}$$

Where used, w_i is the user-provided radius (aka. width) of the place cells (defaulting to 0.2 m).

`GridCells`

Each grid cell is assigned a random wave direction $\theta_i \sim \mathcal{U}_{[0,2\pi]}$, gridscale $\lambda_i \sim \mathcal{U}_{[0.5\text{ m},1.0\text{ m}]}$ and phase offset $\phi_i \sim \mathcal{U}_{[0,2\pi]}$. The firing rate of each grid cell is given by the thresholded sum of three cosines

$$F_i(t) = \frac{1}{3}\max\left(0, \cos\left(2\pi\frac{\mathbf{x}(t)\cdot\mathbf{e}_{\theta_i}}{\lambda_i} + \phi_i\right) + \cos\left(2\pi\frac{\mathbf{x}(t)\cdot\mathbf{e}_{\theta_i+\pi/3}}{\lambda_i} + \phi_i\right) + \cos\left(2\pi\frac{\mathbf{x}(t)\cdot\mathbf{e}_{\theta_i+2\pi/3}}{\lambda_i} + \phi_i\right)\right). \tag{A.31}$$

$\mathbf{e}_\theta$ is the unit vector pointing in the direction θ. A shifted (as opposed to rectified) sum of three cosines grid cell is also provided resulting in softer grid fields

$$F_i(t) = \frac{2}{3}\left(\frac{1}{3}\left(\cos\left(2\pi\frac{\mathbf{x}(t)\cdot\mathbf{e}_{\theta_i}}{\lambda_i} + \phi_i\right) + \cos\left(2\pi\frac{\mathbf{x}(t)\cdot\mathbf{e}_{\theta_i+\pi/3}}{\lambda_i} + \phi_i\right) + \cos\left(2\pi\frac{\mathbf{x}(t)\cdot\mathbf{e}_{\theta_i+2\pi/3}}{\lambda_i} + \phi_i\right)\right) + \frac{1}{2}\right) \tag{A.32}$$

$\mathbf{e}_\theta$ is the unit vector pointing in the direction θ.

`VectorCells` (parent class only)

`VectorCells` subclasses include `BoundaryVectorCells`, `ObjectVectorCells` and `AgentVectorCells` as well as `FieldOfView` versions of these three classes. The common trait amongst all types of `VectorCell` is that each cell is responsive to a *feature* of the environment (boundary segments, objects, other agents) at a preferred distance and angle. The firing rate of each vector cell is given by the product of two functions; a Gaussian radial function and a von Mises angular function. When the agent is a euclidean distance $d(t)$ from the feature, at an angle $\phi(t)$ the contribution of that feature to the total firing rate is given by

$$g_i(r(t), \theta(t)) = \exp\left(-\frac{(d_i - d(t))^2}{2\sigma_{d,i}^2} \right) \cdot f_{\mathrm{VM}}(\phi(t)|\phi_i, \kappa_i) \tag{A.33}$$

where f_{VM} is the radial von Mises distribution (a generalisation of a Gaussian for periodic variables)

$$f_{\mathrm{VM}}(\phi(t)|\phi_i, \kappa_i) := \exp(\kappa_i \cos(\phi(t) - \phi_i)). \tag{A.34}$$

Total firing rate is calculated by summing/integrating these contributions over all features in the `Environment` as described in the following sections. Distance and angular tuning parameters and defined/sampled as follows:

- d_i is the distance tuning of the vector cell. By default $d_i \sim \mathcal{U}_{[0.05\text{ m},0.3\text{ m}]}$
- $\sigma_{d,i}$ is the distance tuning width. By default this increases linearly as a function of d_i: $\sigma_{d,i} = d_i/\beta + \xi$ for constants β and ξ but can be set otherwise. Values are chosen to match those used by Cothi et al. (2020b).
- ϕ_i is the angular tuning of the vector cell. By default $\phi_i \sim \mathcal{U}_{[0^\circ,360^\circ]}$.
- $\sigma_{\phi,i}$ (which defines the von Mises concentration measure $\kappa_i := 1/\sqrt{\sigma_{\phi,i}}$) is the angular tuning width of the vector cell. By default $\sigma_{\phi,i} \sim \mathcal{U}_{[10^\circ,30^\circ]}$.

The asymptotic equivalence between a Gaussian and a von Mises distribution (true for small angular tunings whereby von Mises distributions of concentration parameter κ approach Gaussian distributions of variance $\sigma^2 = 1/\kappa$) means this model is effectively identical to the original boundary vector cell model proposed by Hartley et al. (2000) but with the difference that my vector cells (BVCs included) will not show discontinuities if they have wide angular tunings of order 360°.

All vector cells can be either

- `allocentric` (default): $\phi(t)$ is the angle subtended between the x-direction vector $\mathbf{e}_x = [1, 0]$, and the line between the `Agent` and the feature.
- `egocentric`: $\phi(t)$ is the angle subtended between the heading direction of the agent $\hat{\mathbf{h}}(t)$, and the line between the `Agent` and the feature.

`BoundaryVectorCells` The environmental features which `BoundaryVectorCells` (BVCs) respond to are the boundary segments (walls) of the `Environment`. The total firing rate of of each cell is given by integrating (computationally I use a default value of $d\theta = 2°$ to numerically approximate this integral) the contributions from the nearest line-of-sight boundary segments (walls occluded by other walls are not considered) around the full 2π field-of-view;

$$F_i(t) = K_i \int_0^{2\pi} g_i(r, \theta) d\theta, \tag{A.35}$$

(computationally I use a default value of $d\theta = 2°$ to numerically approximate this integral). $K_i = 1/\max_{\mathbf{x}} F_i(\mathbf{x})$ is a normalisation constant calculated empirically at initialisation such that each BVC has a maximum firing rate (before scaling) of 1.0 Hz.

`ObjectVectorCells` `ObjectVectorCells` (OVCs) respond to objects in the `Environment`. Objects are zero-dimensional and can be added anywhere within the `Environment`, each object, j, comes with a "type" attribute, t_j. Each object vector cell has a tuning type, t_i, and is only responsive to objects of this type.

The total firing rate of of each cell is given by the sum of the contributions from all objects of the correct type in the `Environment`;

$$F_i(t) = \sum_{\text{objects},j \text{ if } t_j=t_i} g_i(r_j(t), \theta_j(t)). \tag{A.36}$$

Since eq. (A.33) has a maximum value of 1 by definition the maximum firing rate of an object vector cell is also 1 Hz (unless multiple objects are closeby) and no normalisation is required.

`AgentVectorCells` `AgentVectorCells` respond to other `Agent`s in the `Environment`. All cells in a given class are selective to the same `Agent`, index j. The firing rate of each cell is then given by;

$$F_i(t) = g_i(r_j(t), \theta_j(t)). \tag{A.37}$$

`FieldOfViewBVCs, FieldOfViewOVCs & FieldOfViewAVCs` are a special case of the above vector cells where the tuning parameters (d_i, $\sigma_{d,i}$, ϕ_i, $\sigma_{\phi,i}$) for a set of `VectorCells` are carefully set so that cells tile a predefined "field of view". By default these cells are `egocentric` and so the field of view (as the name implies) is defined relative to the heading direction of the `Agent`; if the `Agent` turns the field of view turns with it.

Users define the angular and radial extent of the field of view as well as the resolution of the cells which tile it. There is some flexiblity for users to construct complex fields of view but baic API simplifies this process, exposing a few key parameters:

- $r_{\text{fov}} = [r_{\text{fov}}^{\text{min}}, r_{\text{fov}}^{\text{max}}]$ (default [0.02 m, 0.2 m]): the radial extent of the field of view.
- θ_{fov} (default [0°, 75°]): the angular extend of the field of view (measured from the forward heading direction, symmetric left and right).
- δ_{fov}^0 (default 0.02 m): `FieldOfView` `VectorCells` all have approximately circular receptive fields (i.e. the radial Gaussian and angular von Mises in

eq. (A.33) have matched variances which depend on their tuning distance; $\sigma_{d,i} = d_i \cdot \sigma_{\phi,i} := \delta_{\text{fov}}(d_i))$. δ^0_{fov} sets the resolution of the inner-most row of cells in the field of view, $\delta^0_{\text{fov}} = \delta_{\text{fov}}(d_i = r^{\min}_{\text{fov}})$.

- Manifold type: For "diverging" manifolds (default) cells further away from the `Agent` have larger receptive fields $\delta_{\text{fov}}(d_i) = \xi_0 + d_i/\beta$ for user-defined β (default $\beta = 5$) and $\xi_0 := \delta^0_{\text{fov}} - r^{\min}_{\text{fov}}/\beta$. For "uniform" manifold all cells have the same sized receptive fields, $\delta_{\text{fov}}(d_i) = \delta^0_{\text{fov}}$.

More complex field of views can be constructed and a tutorial is provided to show how.

`HeadDirectionCells`

In 2D `Environment`s each head direction cell has an angular tuning mean θ_i and width $\sigma_i := 1/\sqrt{\kappa_i}$. The response function is then a von Mises in the head direction of the `Agent`:

$$F_i(t) = \exp(\kappa_i \cos(\theta_h(t) - \theta_i)). \tag{A.38}$$

By default all cells have the same angular tuning width of 30° and tuning means even spaced from 0° to 360°.

In 1D `Environment`s there is always and only exactly $n = 2$ `HeadDirectionCells`; one for leftward motion and one for rightward motion.

$$\begin{aligned} F_1(t) &= \max(0, \text{sgn}(\mathbf{v}_{1D}(t))) \\ F_2(t) &= \max(0, \text{sgn}(-\mathbf{v}_{1D}(t))) \end{aligned} \tag{A.39}$$

`VelocityCells`

`VelocityCells` are a subclass of `HeadDirectionCells` which encode the full velocity vector rather than the (normalised) head direction. In this sense they are similar to `HeadDirectionCells` but their firing rate will increase with the speed of the `Agent`.

In 2D their firing rate is given by:

$$F_i(t) = \frac{\mathsf{v}_{2D}}{\sigma_\mathsf{v}} \exp(\kappa_i \cos(\theta_\mathsf{v}(t) - \theta_i)) \tag{A.40}$$

where $\theta_\mathsf{v}(t)$ is the angle of the velocity vector $\mathbf{v}(t)$ anticlockwise from the x-direction and σ_v is the likely speed scale of the `Agent` moving under random motion (this is chosen so the firing rate of the velocity cell before scaling is approximately $\mathsf{O}(1)$ Hz).

In 1D environments:

$$\begin{aligned} F_1(t) &= \max\left(0, \frac{\mathsf{v}_{1D}(t)}{\sigma_\mathsf{v} + \mu_\mathsf{v}}\right), \\ F_2(t) &= \max\left(0, -\frac{\mathsf{v}_{1D}(t)}{\sigma_\mathsf{v} + \mu_\mathsf{v}})\right) \end{aligned} \tag{A.41}$$

where the addition of μ_v accounts for any bias in the motion.

`SpeedCell`

A single cell encodes the scaled speed of the `Agent`

$$F(t) = \frac{\|\mathbf{v}(t)\|}{\sigma_\mathsf{v}} \tag{A.42}$$

where, same as with the `VelocityCells`, σ_v (or $\sigma_\mathsf{v} + \mu_\mathsf{v}$ in 1D) is the typical speed scale of the `Agent` moving under random motion giving these cells ad pre-scaled maximum firing rate of $\mathsf{O}(1)$ Hz.

`PhasePrecessingPlaceCells`

`PhasePrecessingPlaceCells` (a subclass of `PlaceCells`) display a phenomena known as phase precession with respect to an underlying theta oscillation; within each theta cycle the firing rate of a place cell peaks at a phase dependent on how far through the place field the `Agent` has travelled. Specifically, as the `Agent` enters the receptive field the firing rate peaks at a late phase in the cycle and as the `Agent` leaves the receptive field the firing rate peaks at an early phase in the cycle, hence the name phase *pre*cession. Phase precession is

implemented by modulating the spatial firing rate of `PlaceCells` with a phase precession factor, $F_i^\theta(t)$,

$$F_i(t) \leftarrow F_i(t) \cdot F_i^\theta(t), \tag{A.43}$$

which rises and falls each theta cycle, according to:

$$F_i^\theta(t) = 2\pi f_{\text{VM}}\left(\phi_\theta(t)\middle|\phi_i^*\left(\mathbf{x}(t), \dot{\mathbf{x}}(t)\right), \kappa_\theta\right). \tag{A.44}$$

This is a von Mises factor where $\phi_\theta(t) = 2\pi\nu_\theta t \mod 2\pi$ is the current phase of the ν_θ Hz theta-rhythm and $\phi_i^*\left(\mathbf{x}(t), \hat{\mathbf{x}}(t)\right)$ is the current 'preferred' theta phase of a cell which is a function of it's position $\mathbf{x}(t)$ and direction of motion $\hat{\mathbf{x}}(t)$. This preferred phase is calculated by first establishing how far through a cells spatial receptive field the `Agent` has travelled along its current direction of motion;

$$d_i(\mathbf{x}(t), \hat{\mathbf{x}}(t)) = (\mathbf{x}(t) - \mathbf{x}_i) \cdot \hat{\mathbf{x}}(t), \tag{A.45}$$

and then mapping this to a uniform fraction β_θ of the range $[0, 2\pi]$;

$$\phi_i^*(t) = \pi - \beta_\theta \pi \frac{d_i(t)}{\sigma_i}. \tag{A.46}$$

σ_i is the width of the cell at its boundary, typically defined as $\sigma_i = w_i$, except for `gaussian` place cells where the boundary is arbitrarily drawn at two standard deviations $\sigma_i = 2w_i$.

The intuition for this formula can be found by considering an `Agent` travelling straight through the midline of a circular 2D place field. As the `Agent` enters into the receptive field (at which point $(\mathbf{x}(t) - \mathbf{x}_i) \cdot \hat{\mathbf{x}}(t) = -\sigma_i$) the firing rate will peak at a theta phase of $\pi + \beta\pi$. This then precesses backwards as it passes through the field until the moment it leaves ($(\mathbf{x}(t) - \mathbf{x}_i) \cdot \hat{\mathbf{x}}(t) = \sigma_i$) when the firing rate peaks at a phase of $\pi - \beta\pi$. This generalises to arbitrary curved paths through 2D receptive fields. This model has been used and validated before by Jeewajee et al. (2014). κ_θ determines the spread of the von

Mises, i.e. how far from the preferred phase the cell is likely to fire.

`RandomSpatialNeurons`

`RandomSpatialNeurons` provide spatially "tuned" inputs for use in instances where `PlaceCells`, `GridCells`, `BoundaryVectorCells` etc. These neurons have smooth but, over long distances, random receptive fields (approximately) generated by sampling from a Gaussian process with a radial basis function kernel of lengthscale l (default $l = 0.1$ m). The kernel is given by:

$$k(\mathbf{x}, \mathbf{x}') = \exp^{-\frac{d(\mathbf{x},\mathbf{x}')^2}{2l^2}} \tag{A.47}$$

where $d(\mathbf{x}, \mathbf{x}')$ is the environmental-distance between two points in the environment. This distance measure (same as used for `PlaceCells`, and `VectorCells` etc.) accounts for walls in the environment and so the receptive fields of these neurons are smooth everywhere except across walls (see Appendix A.2.2).

Firing rates are calculated as follows: At initialisation an array of target locations, at least as dense as the lengthscale, is sampled across the environment $\{\mathbf{x}_j\}$. For each neuron, i, j target *values*, $[\tilde{F}_i]_:$, is sampled from the multivariate Normal distribution

$$[\tilde{F}_i]_: \sim \mathcal{N}(\mathbf{0}, \mathbf{K}) \tag{A.48}$$

where $\mathbf{K}$ is the covariance matrix with elements $K_{lm} = k(\mathbf{x}_l, \mathbf{x}_m)$. This creates a sparse set of locations, $\{\mathbf{x}_j\}$, and targets, $\tilde{F}_{ij}$, across the `Environment`: locations close to each other are likely to have similar targets (and hence similar firing rates) whereas locations far apart will be uncorrelated.

At inference time the firing rate at an arbitrary position in the `Environment`, $\mathbf{x}(t)$ (which will not neccesarily be one of the pre-sampled targets) is estimated by taking the mean of the targets weighted by the kernel function between the position and the target location:

$$F_i(\mathbf{x}(t)) = \frac{\sum_j k(\mathbf{x}(t), x_j)\tilde{F}_{i,j}}{\sum_j k(\mathbf{x}(t), x_j)} \tag{A.49}$$

This weighted average is a cheap and fast approximation to the true Bayesian Gaussian process which would require the inversion of the covariance matrix $\mathbf{K}$ at each time-step and which I find to be numerically unstable around exposed walls.

`FeedForwardLayer`

`FeedForwardLayer` and `NeuralNetworkNeurons` are different from other RatInABox classes; their firing rates are not explicitly determined by properties (position, velocity, head direction etc.) of their `Agent` but by the firing rates of a set of input layers (other `ratinabox.Neurons`). They allow users to create arbitrary and trainable "function approximator" `Neurons` with receptive fields depending non-trivially on the states of one or many `Agent`(s).

Each `FeedForwardLayer` has a list of inputs $\{\mathsf{L}_j\}_{j=1}^N$ which must be other `ratinabox.Neurons` subclasses (e.g. `PlaceCells`, `BoundaryVectorCells`, `FeedForwardLayer`). For input layer j with n_j neurons of firing rates $F_k^{\mathsf{L}_j}(t)$ for $k \in [1, n_j]$, a weight matrix is initialised by drawing weights randomly $\mathsf{w}_{ik}^{\mathsf{L}_j} \sim \mathcal{N}(0, g/\sqrt{n_j})$ (for default weight initialisation scale $g = 1$). The firing rate of the i^{th} `FeedForwardLayer` neuron is given by weighted summation of the inputs from all layers plus a bias term:

$$r_i(t) = \sum_{j=1}^{N} \sum_{k=1}^{n_j} \mathsf{w}_{ik}^{\mathsf{L}_j} F_k^{\mathsf{L}_j}(t) + b_i \tag{A.50}$$

$$F_i(t) = \phi(r_i(t)) \tag{A.51}$$

where $\phi(x)$ is a potentially non-linear activation function defaulting to a linear identity function of unit gain. b_i is a constant bias (default zero). A full list of available activations and their defining parameters can be found in the `utils.py` file; these include ReLU, sigmoid, tanh, Retanh, softmax and linear (the default) functions or users can pass their own bespoke activation function.

Alongside $\phi(r_i(t))$ this layer also calculates and saves $\phi'(r_i(t))$ where ϕ' is the derivative of the activation function, a necessary quantity for many learning rules and training algorithms.

NeuralNetworkNeurons

`NeuralNetworkNeurons` are a generalisation of `FeedForwardLayer`. Like `FeedForwardLayer` they are initialised with a list of inputs $\{\mathsf{L}_j\}_{j=1}^{N}$. This class also receives, at the point of initialisation, a neural network, NN. This can be any `pytorch.nn.module`. To calculate the firing rate this class takes the firing rates of all input layers, concatenates them, and passes them through the neural network. The firing rate of the i^{th} `NeuralNetworkNeurons` neuron is given by the activity of the i^{th} neuron in the output layer of neural network:

$$F_i(t) = \mathsf{NN}_i(\underbrace{\vec{F}^{\mathsf{L}_1}(t), \vec{F}^{\mathsf{L}_2}(t), ...;}_{\text{inputs}} \underbrace{\mathsf{w}}_{\text{weights}}) \tag{A.52}$$

If no neural network is provided by the user a default network with two hidden ReLU layers of size 20 is used.

In order to be compatible with the rest of the RatInABox API the firing rate returned by this class is a `numpy` array, however, on each update the output of the `pytorch` neural network is additionally saved as a `torch` tensor. By accessing this tensor, users can take gradients back through the embedded neural network and train is as I demonstrate in Figure 1.3e.

In Figure 1.3e and an associated demo script a `NeuralNetworkNeurons` layer is initialised with $N = 1$ neuron/output. The inputs to the network come from a layer of 200 `GridCells`, ranging in grid scale from 0.2 m to 0.5 m. These are passed through a neural network with three hidden ReLU layers of size 100 and a linear readout. As the `Agent` randomly explores its `Environment` the network is trained with gradient descent to reduce the L2 error between the firing rate of the network and that of a "target" rate map (a vector image of the letters "RIAB"). I use gradient descent with momentum and a learning rate of $\eta = 0.002 \cdot \mathrm{dt}^2$ (which makes the total rate of learning time-step independent). Momentum is set to $\mu = (1 - \frac{dt}{\tau_{\text{et}}})$ where τ_{et} is the eligibility trace timescale of 10 seconds which smoothes the gradient descent, improving convergence. I find learning converges after approximately 2 hours and a good approximation of

the target function is achieved.

A.2.4 Table of Default Parameters

This table lists the RatInABox parameters and their default values. The 'Key' column gives the key in a parameters dictionary which can be passed to each class upon initialisation. Any variables not present in the parameters dictionary at initialisation will be taken as default. For example, initialising an `Environment` of size 2 m (which is *not* the default size) and adding an `Agent` with a mean speed of 0.3 ms^{-1} (which is *not* the default size) would be done as follows:

```
import ratinabox
from ratinabox.Environment import Environment
from ratinabox.Agent import Agent

Env = Environment(params = {"scale":2.0})   # initialise non-
                                    default Environment
Ag = Agent(Env, params = {"speed_mean":0.3}) # initialise non-
                                    default Agent
```

Table A.1: Default values, keys, and allowed ranges for `ratinabox` parameters.
* Passed as a `kwarg` to `Agent.update()`.
** Passed as a `kwarg` to `FeedForwardLayer.add_input()`.

Parameter	Key	Description (unit)	Default	Acceptable range
`Environment()`				
D	`dimensionality`	Dimensionality of the `Environment`.	`"2D"`	[`"1D"`, `"2D"`]
Boundary conditions	`boundary_conditions`	Determines agent behavior at boundaries.	`"solid"`	[`"solid"`, `"periodic"`]
Scale, s	`scale`	Size of the environment (m).	1.0	$\mathbb{R}^+$
Aspect ratio, a	`aspect`	Aspect ratio for 2D environments; width $= sa$, height $= s$.	1.0	$\mathbb{R}^+$
dx	`dx`	Discretization for plotting rate maps (m).	0.01	$\mathbb{R}^+$

Continued on next page...

Parameter	Key	Description (unit)	Default	Acceptable range
Walls	`walls`	List of internal walls (m). Typically added via `Env.add_wall()`.	`[]`	$N_{\text{walls}} \times 2 \times 2$ array/list
Boundary	`boundary`	List of coordinates for non-rectangular environment perimeters (m).	`None`	$N_{\text{corners}} \times 2$ array/list
Holes	`holes`	List of coordinate lists, each bounding a hole (m).	`None`	$N_{\text{holes}} \times \geq 3 \times 2$ array/list
Objects	`objects`	List of objects (m). Typically added via `Env.add_object()`.	`[]`	$N_{\text{objects}} \times 2$ array/list
`Agent()`				
dt	`dt`	Time discretisation step size (s).	0.01	$\mathbb{R}^+$

Continued on next page...

Parameter	Key	Description (unit)	Default	Acceptable range
τ_v	`speed_coherence_time`	Timescale for speed decoherence in random motion (s).	0.7	$\mathbb{R}^+$
σ_v (2D) μ_v (1D)	`speed_mean`	2D: Rayleigh scale for random motion. 1D: Normal mean for random motion (ms^{-1}).	0.08	2D: $\mathbb{R}^+$ 1D: $\mathbb{R}$
σ_v	`speed_std`	Normal std dev for random motion in 1D (ms^{-1}).	0.08	$\mathbb{R}^+$
τ_ω	`rotational_velocity_coherence_time`	Rotational velocity decoherence timescale (s).	0.08	$\mathbb{R}^+$
σ_ω	`rotational_velocity_std`	Rotational velocity Normal std dev (rad s^{-1}).	$2\pi/3$	$\mathbb{R}^+$
λ_{thig}	`thigmotaxis`	Thigmotaxis parameter.	0.5	$0 < \lambda_{\text{thig}} < 1$
d_{wall}	`wall_repel_distance`	Wall range of influence (m).	0.1	$\mathbb{R}^+$
s	`walls_repel_strength`	Strength of wall repulsion.	1.0	$\mathbb{R}_0^+$

Continued on next page...

Parameter	Key	Description (unit)	Default	Acceptable range
k	`drift_to_random_strength_ratio`*	Ratio of drift velocity to random motion.	1.0	$\mathbb{R}_0^+$
`Neurons()`				
n	`n`	Number of neurons.	10	$\mathbb{Z}^+$
$f_{\max}$	`max_fr`	Maximum firing rate (Hz).	1.0	$\mathbb{R}$
$f_{\min}$	`min_fr`	Minimum firing rate (Hz).	0.0	$f_{\min} < f_{\max}$
σ_η	`noise_std`	Std dev of OU noise added to firing rates (Hz).	0.0	$\mathbb{R}^+$
τ_η	`noise_coherence_time`	Timescale of OU noise (s).	0.5	$\mathbb{R}^+$
Name	`name`	A name to identify the `Neurons` instance.	`"Neurons"`	Any string
`PlaceCells()`				

Continued on next page...

Parameter	Key	Description (unit)	Default	Acceptable range
Type	`description`	Place cell firing function.	`"gaussian"`	[`"gaussian"`, `"gaussian_threshold"`, `"diff_of_gaussians"`, `"top_hat"`, `"one_hot"`]
w_i	`widths`	Place cell width (m). Can be a single number or an array.	0.2	$\mathbb{R}^+$
$\{\mathbf{x}_i^{\text{PC}}\}$	`place_cell_centres`	Place cell locations (m). If `None`, cells are randomly scattered.	`None`	`None` or array of positions
Wall geometry	`wall_geometry`	How place cells interact with walls.	`"geodesic"`	[`"geodesic"`, `"line_of_sight"`, `"euclidean"`]
`GridCells()`				

Continued on next page...

Parameter	Key	Description (unit)	Default	Acceptable range
λ_i	`gridscale`	Grid scales (m), or parameters for sampling distribution.	(0.5,1)	array-like or tuple
λ_i-dist	`gridscale_distribution`	Distribution for sampling grid scales if not provided manually.	`"uniform"`	See `utils.distribution_sampler()`
θ_i	`orientation`	Orientations (rad), or parameters for sampling distribution.	(0,2π)	array-like or tuple
θ_i-dist	`orientation_distribution`	Distribution for sampling orientations if not provided manually.	`"uniform"`	See `utils.distribution_sampler()`
ϕ_i	`phase_offset`	Phase offsets (rad), or parameters for sampling distribution.	(0,2π)	array-like or tuple

Continued on next page...

Parameter	Key	Description (unit)	Default	Acceptable range
ϕ_i-dist	`phase_offset_distribution`	Distribution for sampling phase offsets if not provided manually.	`"uniform"`	See `utils.distribution_sampler()`
Type	`description`	Grid cell firing function.	`"three_rectified_cosines"`	[`"three_rectified_cosines"`, `"three_shifted_cosines"`]
`VectorCells()`				
Reference frame	`reference_frame`	Allo- or egocentric coordinate frame for receptive fields.	`"allocentric"`	[`"allocentric"`, `"egocentric"`]
Arrangement protocol	`cell_arrangement`	How receptive fields are arranged in the environment.	`"random"`	[`"random"`, `"uniform_manifold"`, `"diverging_manifold"`, function()]

Continued on next page...

Parameter	Key	Description (unit)	Default	Acceptable range
d_i	`tuning_distance`	Tuning distances (m), or params for sampling dist.	(0.0,0.3)	array-like or tuple
d_i-dist	`tuning_distance_distribution`	Dist. for sampling tuning distances if not provided manually.	`"uniform"`	See `utils.distribution_sampler()`
$\sigma_{d,i}$	`sigma_distance`	Distance tuning widths (m), or params for dist.	(0.08,12)	array-like or tuple
$\sigma_{d,i}$-dist	`sigma_distance_distribution`	Dist. for sampling distance tuning widths. `"diverging"` is a special case.	`"diverging"`	See `utils.distribution_sampler()`
ϕ_i	`tuning_angle`	Tuning angles (°), or params for sampling dist.	(0.0,360.0)	array-like or tuple
ϕ_i-dist	`tuning_angle_distribution`	Dist. for sampling tuning angles if not provided manually.	`"uniform"`	See `utils.distribution_sampler()`

Continued on next page...

Parameter	Key	Description (unit)	Default	Acceptable range
$\sigma_{\phi,i}$	`sigma_angle`	Angular tuning widths (°), or params for dist.	(10,30)	array-like or tuple
$\sigma_{\phi,i}$-dist	`sigma_angle_distribution`	Dist. for sampling angular tuning widths if not provided manually.	`"uniform"`	See `utils.distribution_sampler()`
`BoundaryVectorCells()`				
$d\theta$	`dtheta`	Size of angular integration step (°).	2.0	$0 < d\theta \ll 360$
`ObjectVectorCells()`				
t_i	`object_tuning_type`	Tuning type for object vectors.	`"random"`	`"random"`, `int`, or array-like
wall-behaviour	`walls_occlude`	Whether walls occlude objects behind them.	True	`bool`
`AgentVectorCells()`				

Continued on next page...

Parameter	Key	Description (unit)	Default	Acceptable range
Other agent, j	`Other_Agent`	The `Agent` these cells are selective for.	`None`	`ratinabox.Agent`
wall-behaviour	`walls_occlude`	Whether walls occlude other `Agent`s.	True	`bool`
`FieldOfView[X]s()` for `[X]` in [`BVC`,`OVC`,`AVC`]				
r_{fov}	`distance_range`	Radial extent of the field-of-view (m).	[0.02,0.4]	List of two distances
θ_{fov}	`angle_range`	Angular range of the field-of-view (°).	[0,75]	List of two angles
δ^0_{fov}	`spatial_resolution`	Resolution of the innermost row of vector cells (m).	0.02	$\mathbb{R}^+$
β	`beta`	Inverse gradient for receptive field size increase with distance.	5	$\mathbb{R}^+$

Continued on next page...

Parameter	Key	Description (unit)	Default	Acceptable range
Arrangement protocol	`cell_arrangement`	How the field-of-view receptive fields are constructed.	`"diverging_manifold"`	[`"diverging_manifold"`, `"uniform_manifold"`]
`FeedForwardLayer()`				
$\{\mathsf{L}_j\}_{j=1}^{N}$	`input_layers`	A list of upstream `Neurons` classes.	[]	N-list of `Neurons`
Activation function	`activation_function`	Defines the activation function, either by name or as a custom function.	{`"activation"`: `"linear"`}	See `utils.activate()`
g	`w_init_scale`**	Scale of random weight initialisation.	1.0	$\mathbb{R}^+$
b_i	`biases`	Biases, one per neuron (optional).	$[0, ..., 0]$	$\mathbb{R}^n$
`NeuralNetworkNeurons()`				

Continued on next page...

Parameter	Key	Description (unit)	Default	Acceptable range
$\{\mathsf{L}_j\}_{j=1}^N$	`input_layers`	A list of upstream `Neurons` classes.	[]	A list of `Neurons`
NN	`NeuralNetworkModule`	Internal neural network mapping inputs to outputs. A default is used if `None`.	`None`	Any `torch.nn.Module`
`RandomSpatialNeurons()`				
l	`lengthscale`	Lengthscale of the Gaussian process kernel (m).	0.1	$\mathbb{R}^+$
Wall geometry	`wall_geometry`	How distances are calculated (and thus how cells interact with walls).	`"geodesic"`	[`"geodesic"`, `"line_of_sight"`, `"euclidean"`]
`PhasePrecessingPlaceCells()`				
ν_θ	`theta_freq`	The theta frequency (Hz).	10.0	$\mathbb{R}^+$
κ_θ	`kappa`	The phase precession breadth parameter.	1.0	$\mathbb{R}^+$

Continued on next page...

Parameter	Key	Description (unit)	Default	Acceptable range
β_θ	`beta`	The phase precession fraction.	0.5	$0.0 < \beta < 1.0$

A.2.5 Tutorials and Demos

Numerous resources are provided, some of which are listed here, to streamline the process of learning RatInABox. Next to each, the key features – which you may be interested in learning – covered by the resource are described.

- **Github readme:** Installing and importing RatInABox. Descriptions and diagrams of key features.
- **Simple script:** A minimal example of using RatInABox to generate and display data. Code duplicated below for convenience.
- **Extensive script:** A more detailed tutorial showing advanced data generation, and advanced plotting.
- **Decoding position example:** Data collection. Firing rate to position decoding. Data plotting.
- **Conjunctive cells example:** `GridCells` and `HeadDirectionCells` are combined with the function approximator `FeedForwardLayer` class to make head direction-selective grid cells (aka. conjunctive grid cells)
- **Splitter cells example:** Bespoke `Environment`, `Agent` and `Neurons` subclasses are written to make simple model of splitter cells.
- **Successor features example:** Loop-shaped `Environment` is constructed. Implementation of TD learning.
- **Reinforcement Learning Example**: A bespoke `ValueNeuron` subclass is defined. Implementation of TD learning. External 'non-random' control of `Agent` velocity.
- **Deep learning example**: Deep `NeuralNetworkNeurons` trained to approximate a target function. Bespoke `Neurons` subclass encoding a `.png` is written.

- **Actor-critic example**: Deep `NeuralNetworkNeurons` are used to implement the actor-critic algorithm in egocentric and allocentric action/representation spaces.
- **Path Integration Example**: Extensive use of `FeedForwardLayer` to build a deep multilayer network. Implementation of a local Hebbian learning rule.
- **List of plotting functions:** Lists and describes all available plotting functions.

In addition, scripts reproducing all figures in the GitHub readme and this chapter are provided too. The code comments are nearly comprehensive and can be referenced for additional understanding where needed.

A simple script

See the GitHub repository for instructions on how to install RatInABox. The following is a Python script demonstrating a very basic use-case.

Import RatInABox and necessary classes. Initialise a 2D `Environment`. Initialise an `Agent` in the `Environment`. Initialise some `PlaceCells`. Simulate for 20 seconds. Print table of times, position and firing rates. Plot the motion trajectory, the firing rate timeseries' and place cell rate maps.

```
# Import RatInABox
import ratinabox
from ratinabox.Environment import Environment
from ratinabox.Agent import Agent
from ratinabox.Neurons import PlaceCells
import pandas as pd

# Run a very simple simulation
Env = Environment()
Ag = Agent(Env)
PCs = PlaceCells(Ag)
for i in range(int(20/Ag.dt)):
  Ag.update()
```

```
  PCs.update()

# Export data into a dataframe
pd.DataFrame(Ag.history)

# Plot data
Ag.plot_trajectory()
PCs.plot_rate_timeseries()
PCs.plot_rate_map()
```

A.2.6 License Information

RatInABox is currently distributed under an MIT License, meaning users are permitted to use, copy, modify, merge, publish, distribute, sublicense and sell copies of the software.

A.3 Demonstrations and Use Cases

A.3.1 Figure Details and Parameters

A Jupyter script replicating figs. 1.1 to 1.3 can be found at https://github.com/RatInABox-Lab/RatInABox/blob/main/demos/paper_-figures.ipynb.

Figure 1.1: Panel (b): Place cells are of type `gaussian_threshold` with widths $w_i = 0.4$ m. Panel (e) $\mu_v = 0.1$ and $\sigma_v = 0.2$.

Figure 1.2: Panel (a): Curve fitting is done using `scipy.optimize.curve_fit`. Panel (d): $dt = 100$ ms. Panel (e) `Agent.wall_repel_strength` $= 2$. Panel (e) uses *all* available datasets from Sargolini et al. (2006) to create the histograms, as opposed to panel (a) which only uses one of the recordings.

Figure 1.3: Panel (a): 25 seconds of trajectory data from Sargolini et al. (2006) is imported, converted into metres, mean centred and then downsampled by 30x (from 50 Hz to 1.66 Hz) before being imported into a RatInABox `Agent`. Panel (c): All populations of vector cells had `"distance_range"` = [0.05, 0.30], `"angle_range"` = [0,75] and `"spatial_resolution"` = 0.015. Panel (e):

RatInABox for reinforcement learning experiment is described below. Panel (f): The average and standard deviation over 1000 repeats is plotted. For the motion model this is taken for motion updates of a default `Agent` in a default `Environment` (i.e. 2D with solid boundary conditions and no additional walls). For the `numpy` matrix calculations the time taken does not include the time taken to initialise the matrices.

A.3.2 Supplementary Use Cases

In this section it is demonstrated how RatInABox could be used in two simple experiments: neural decoding and reinforcement learning. The intention is not to present novel scientific results but rather to demonstrate the capability of RatInABox to facilitate original scientific research in a variety of fields. Additional demos beyond these two are given in the online repository and, as with all figures in this chapter, executable Jupyter scripts are provided to replicate all figures shown.

Neural Decoding Example

Jupyter script: `https://github.com/RatInABox-Lab/RatInABox/blob/main/demos/paper_figures.ipynb`.

This demonstration studies, using RatInABox, which type of spatially modulated cell type is best for decoding position.

Training and testing datasets are first generated. A set of `Neurons` ($n = N_{\text{cells}} = 20$) is initialised in a 1 m square `Environment` containing a small barrier (Figure A.1, top). A six minute trajectory is simulated using the RatInABox random motion model to produce a dataset of inputs $\{\mathbf{x}_i\}_{i=1}^{N_T}$ and targets $\{\mathbf{y}_i\}_{i=1}^{N_T}$:

$$\mathbf{x}_i = \vec{F}(\mathbf{x}(t_i)) \sim \mathcal{X} \subseteq \mathbb{R}^{N_{\text{cells}}} \tag{A.53}$$

$$\mathbf{y}_i = \mathbf{x}(t_i) \sim \mathcal{Y} \subseteq \mathbb{R}^2 \tag{A.54}$$

where $\mathbf{x}(t_i)$ is the position of the `Agent` at time t_i and $\vec{F}$ is the firing rate of the neuronal population. These data are split into training ($0 < t_i < 5$ mins,

fig. A.1a purple) and testing ($5 < t_i < 6$ mins, fig. A.1a black) fractions. The goal of the decoder is to learn a mapping $G : \mathcal{X} \to \mathcal{Y}$, from firing rates to positions.

To do this I use Gaussian Process Regression (GPR). GPR is a form of non-parameteric regression where a prior is placed over the infinite-dimensional function space $P(G(\mathbf{x}))$ in the form of its covariance kernel $C(\mathbf{x}, \mathbf{x}')$ and mean $\mu(\mathbf{x})$ (typically zero). This defines a prior on the targets in the training set $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, ...)^\mathsf{T}$,

$$P(\mathbf{Y}) = \mathcal{N}(\mathbf{Y}; \mathbf{0}, \mathsf{C}), \tag{A.55}$$

where $\mathsf{C}_{ij} = C(\mathbf{x}_i, \mathbf{x}_j) + \sigma_\eta \delta_{ij}$ is a covariance matrix established over the data points. The second term accounts for additive noise in the data function. This can be used to make an inference on the posterior of the target for an unseen testing data point, $P(\mathbf{y}_{\text{test}} | \{\mathbf{x}_i\}^{\text{train}}, \{\mathbf{y}_i\}^{\text{train}}, \mathbf{x}_{\text{test}})$ – itself a Gaussian – the mean of which is taken as the "prediction". A more comprehensive reference/tutorial on Gaussian Process Regression is given by MacKay (2003).

I use a radial basis function (aka "squared exponential") kernel with width $l = l_0\sqrt{N_{\text{cells}}}$ which scales with the expected size of the population vector ($\sim \sqrt{N_{\text{cells}}}$, I set $l_0 = \sqrt{20}$)

$$C(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2l^2}\right) \tag{A.56}$$

and a small amount of target noise $\sigma_\eta = 1e - 10$. Note that the closest 'parameterised' analog to GPR with an RBF kernel is linear regression against Gaussian basis features of length scale l. Since the Gaussian is a non-linear function this means my regression prior is also a non-linear function of firing rate (and therefore potential non-biologically plausible). I choose to optimise with the `sklearn.gaussian_process.GaussianProcessRegressor` package. Note I do not attempt to optimise the hyperparameters l_0 or σ_η which one would probably do in a more rigorous experiment. RatInABox parameters are all default with the exception that the place cells are of type `gaussian_threshold`

and width $w_i = 0.4$ m and the timestep is set to $dt = 50$ ms.

Figure A.1, panel b (lower) shows the results over comparable sets of `PlaceCells`, `GridCells` and `BoundaryVectorCells`. Coloured dots show the prediction – mean of the posterior – of the GPR model "trained" on all points in the training dataset for that particular cell type. This is plotted on top of the true trajectory, shown in black. `PlaceCells` perform best achieving under 1 cm average decoding error, followed by `BoundaryVectorCells` then `GridCells` where the decoded position is visibly noisy.

Place cells outperform grid cells which outperform BVCs irrespective of how many cells are used in the basis feature set. More cells gives lower decoding error. Decoding errors in Figure A.1c are smaller than would be expected if one decoded from equivalently sized populations of *real* hippocampal neurons. There are likely many reasons for this. Real neurons are noisy, communicate sparsely through spikes rather than rates and, most likely, jointly encode position and many other behaviourally relevant (or irrelevant) variables simultaneously. All of these factors could be straightforwardly incorporated into this analysis using existing RatInABox functionality.

Reinforcement Learning Example

Jupyter script: `https://github.com/RatInABox-Lab/RatInABox/blob/main/demos/reinforcement_learning_example.ipynb`.

In this example it is demonstrated how RatInABox can be used in a reinforcement learning (RL) study. The goal is as follows: train an artificial `Agent` to explore a 2D `Environment` where a reward is hidden behind a wall. The `Agent` should become proficient at navigating around the wall and towards the reward from all locations within the `Environment`.

The core of the approach will rest on model-free RL where an `Agent` first learns a value function for a policy (a process known as "policy evaluation") and then uses this value function to define a new, improved policy ("policy improvement"). Iterating between these two procedures ("policy iteration") can result in convergence towards an optimal or near-optimal policy.

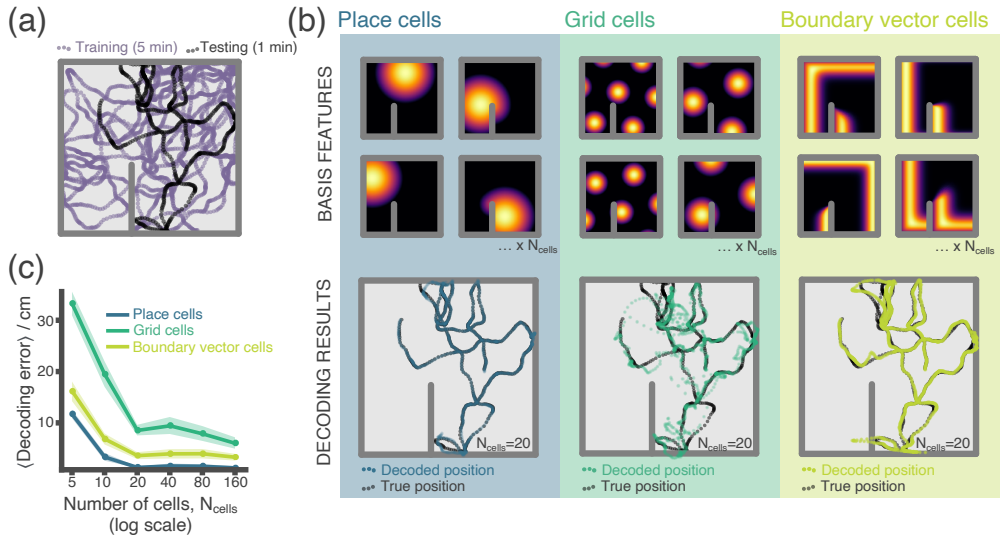

Figure A.1: RatInABox used for a simple neural decoding experiment. **(a)** Training (5 min) and testing (1 min) trajectories are sampled in a 1 m square environment containing a small barrier. **(b)** The firing rates of a population of $N_{\text{cells}} = 20$ cells, taken over the training trajectory, are used to fit a Gaussian Process regressor model estimating position. This decoder is then used to decode position from firing rates on the the unseen testing dataset. Top row shows receptive field for 4 of the 20 cells, bottom row shows decoding estimate (coloured dots) against ground truth (black dots). The process is carried out independently for populations of place cells (left), grid cells (middle) and boundary vector cells (right). **(c)** Average decoding error against number of cells, note log scale. Error region shows the standard error in the mean over 15 random seeds. A Jupyter script demonstrating this experiment is given in the codebase GitHub repository.

A core pillar of RatInABox is its continuous approach to modelling time and space. This continuity will require revising typical approaches to how the value function is defined, approximated and then learned, as well as how motion control (aka action selection, in discrete space) is performed. This is not a weakness, in fact I would argue it is one of the strengths. Once complete, the result is a formulation of model-free RL which bears much higher resemblance to biological navigation. Furthermore, since most of the complexities of feature encoding and motion control in continuous time and space are handled by RatInABox innately this "upgrade" comes almost for free.

Policy evaluation The value of a motion policy, π, is defined as the decaying sum (or integral in continuous time) of expected future rewards

$$\hat{\mathsf{V}}(t) \approx \mathsf{V}^{\pi}(t) = \mathbb{E}\left[\frac{1}{\tau}\int_{t}^{\infty} e^{-\frac{t'-t}{\tau}}\mathsf{R}(t')dt'\right] \tag{A.57}$$

where the expectation is taken over any stochasticity present in the current policy (i.e. how the `Agent` moves) and `Environment`/reward (although in this case both will be deterministic). This definition of value is temporally continuous. The key differences compared to the more common form – where value is written as a discrete sum of rewards over future timesteps – is that it is now a continuous integral over a reward density function and temporal discounting is done by exponentially decaying future reward over a time period τ. The prefactor of $1/\tau$ is an optional constant of normalisation.

In order to learn the value function, a new `ValueNeuron` class is defined. The `ValueNeuron`, which is a subclass of `FeedForwardLayer`, receives feedforward input from a set of features corresponding to `PlaceCells` scattered across the `Environment` with firing rates $\{\phi_i\}_{i=1}^{N_\phi=1000}$ where $\phi_i(\mathbf{x}) = F_i(\mathbf{x})$ is the firing rate of the i^{th} place cell at location $\mathbf{x}$. This linear approximation to the value function can be written as

$$\hat{\mathsf{V}}(\mathsf{I}(t);\mathsf{w}) = \sum_{i=1}^{N} \mathsf{w}_i\phi_i(t). \tag{A.58}$$

I can take the temporal derivative of eq. (A.57) and derive a consistency equation (analogous to the Bellman equation) satisfied by this value function. This naturally gives a temporal difference-style update rule which relies on "bootstrapping" (the current estimate of the value function is used in lieu of the true value function) to optimize the weights of the value function approximation. A good reference for continuous RL is Doya (2000) if readers wish to know more about deriving this learning rule.

$$\delta\mathsf{w}_\mathsf{i}(t) = \eta\left(\mathsf{R}(t) + \tau\frac{d\hat{\mathsf{V}}(t)}{dt} - \hat{\mathsf{V}}(t)\right)\mathsf{e}_i(t). \tag{A.59}$$

For now it suffices to observe that this learning rule is very similar to the temporally discrete TD-update rule. The first term in brackets represents the continuous analog of the temporal difference error (in fact, if you rediscretise using the Euler formula $\dot{\mathsf{V}}(t) = \frac{\mathsf{V}(t+dt)-\mathsf{V}(t)}{dt}$ to replace the derivative, and set $dt = 1$, you will see they are identical). The second term is the 'eligibility trace' determining to which state – or basis feature – credit for the TD error should be assigned. Using an eligibility trace is optional, and it could just be replaced with $\mathfrak{E}_i(t)$, however doing so aids stability of the learning. It is defined as:

$$\mathsf{e}_i(t) = \frac{1}{\tau_\mathsf{e}} \int_{-\infty}^{t} e^{-\frac{t-t'}{\tau_\mathsf{e}}} \mathfrak{E}_i(t') dt'. \tag{A.60}$$

In total the newly defined `ValueNeuron` does three things, schematically laid out in Figure A.2a:

1. It linearly summates its `PlaceCell` inputs, eq. (A.58).
2. It stores and updates the eligibility traces, eq. (A.60).
3. It implements the learning rule, eq. (A.59), which requires access to the reward density function $\mathsf{R}(t)$, the eligibility traces $\mathsf{e}_i(t)$, its firing rate $\hat{\mathsf{V}}(t)$ and the temporal derivative of its firing rate $\frac{d\hat{\mathsf{V}}(t)}{dt}$.

I use a temporal discount horizon of $\tau = 10$ s and an eligibility trace timescale of $\tau_\mathsf{e} = 5$ s. Input features are a set of $N_\phi = 1000$ `PlaceCells` of random widths uniformly sampled from 0.04 m to 0.4 m (Figure A.2b). The reward density function is taken to be the firing rate of a single `PlaceCell` positioned behind the wall of type `top_hat` and width 0.2 m (Figure A.2c). The learning rate is set to $\eta = 1e-4$.

Policy improvement The model now has a neuron capable of learning the value function under its current policy ("policy evaluation"). This is then used to improve the policy ("policy improvement") towards an optimal one. To do this, the "drift velocity" feature is exploited (see Appendix A.2.1). I set the drift velocity to be 3 times the mean velocity in the direction of steepest ascent

of the value function.

$$\mathbf{v}_{\text{drift}}(t) = 3\sigma_{\mathbf{v}}\hat{\nabla}_{\mathbf{x}}\hat{\mathrm{V}}(\mathbf{x}(t)). \tag{A.61}$$

This way the `Agent` is encouraged to move towards regions of higher and higher value. Note that calculating this gradient is a local calculation and can be done on-the-fly by the `Agent`, as it locomotes. This method of value ascent is essentially a continuous analog of a similar algorithm, "greedy policy optimization", used in discrete action spaces.

Policy iteration Learning is done in batches of 8 episodes each. An episode consists of the `Agent` being reset to a random location in the `Environment` and left to explore. The episode ends when the `Agent` gets close to the reward *or* times out (60 seconds). At the start of each batch the current value function is copied and cached - this cached version is used, but not updated, to determine the drift velocity in eq. (A.61) for the duration of the next batch. Varying the strength of the drift bias relative to the random motion allows for control of the trade of between exploration and exploitation. Scheduling goes as follows: initially the `drift_to_random_strength_ratio` is set to $k = 0.1$ (i.e. mostly random exploration). On each successful episode which did not end in a timeout, this is increased by 10% up to a maximum of $k = 1$ (approximately equal contributions of random and drift motions).

Results Initially the input weights to the `ValueNeuron` are drawn randomly $\mathtt{w}_i \sim \mathcal{N}(0, \frac{1}{\sqrt{N_\phi}})$ and therefore the value map (and `Agent` motion) is random (Figure A.2d, left). After 10 batches (80 episodes) the `Agent` has successfully learnt a near-optimal value function showing high-value in and near to the corridor, and low values elsewhere. This allows it to rapidly navigate towards the reward, avoiding the obstructing wall, from all locations in the `Environment` (Figure A.2d, middle, and Figure 1.3b).

By virtue of using continuous action control in continuous space, the trajectories of the trained `Agent` look highly realistic compared to typical gridworld RL. Since `PlaceCells` in RatInABox interact adaptively with the

`Environment`, when a small gap is created at the base of the obstructing wall the receptive fields of `PlaceCells` near this gap "spill" through. This causes an instantaneous update to the perceived value function and therefore policy allowing the `Agent` to immediately find a short cut to the reward with no additional training, a form of zero-shot learning (Figure A.2d, right).

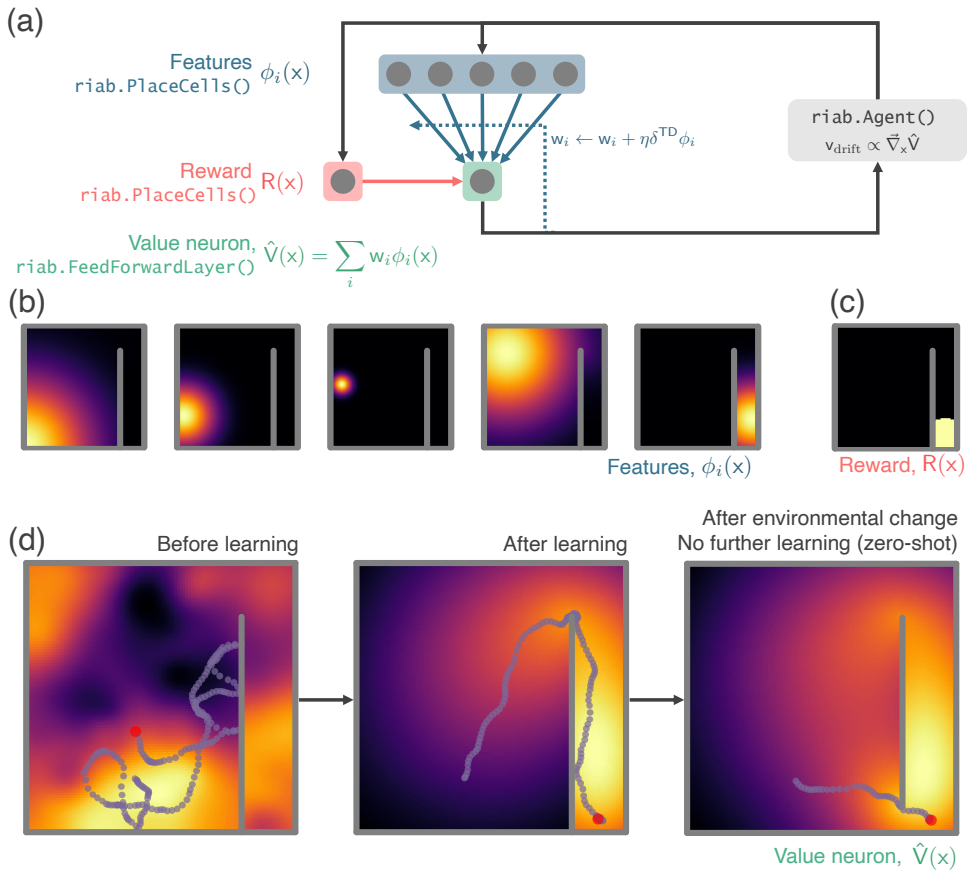

Figure A.2: RatInABox used in a simple reinforcement learning project. **(a)** A schematic of the 1 layer linear network. Using a simple model-free policy iteration algorithm the `Agent`, initially moving under a random motion policy, learns to approach an optimal policy for finding a reward behind a wall. The policy iteration algorithm alternates between (left) calculating the value function using temporally continuous TD learning and (right) using this to define an improved policy by setting the drift velocity of the `Agent` to be proportional to the gradient of the value function (a roughly continuous analog for the ϵ-greedy algorithm). **(b)** 1000 `PlaceCells` act as a continuous feature basis for learning the value function. **(c)** The reward is also a (top-hat) `PlaceCell`, hidden behind the obstructing wall. **(d)** A `ValueNeuron` (a bespoke `Neurons` subclass defined for this demonstration) estimates the policy value function as a linear combination of the basis features (heatmap) and improves this using TD learning. After learning the `Agent` is able to accurately navigate around the wall towards the reward (middle). Because `PlaceCells` in RatInABox are continuous and interact adaptively with the `Environment` when a small gap is opened in the wall place fields corresponding to place cells near this gap automatically bleed through it, and therefore so does the value function. This allows the `Agent` to find a shortcut to the reward with zero additional training. A Jupyter script replicating this project is given in the demos folder GitHub repository.

Appendix B

Appendix to Chapter 2

B.1 Code Availability

Code to generate the results in Chapter 2 is available at `https://github.com/TomGeorge1234/ThetaSequencesAreEligibilityTraces`.

B.2 Task Formulation and Temporal Difference Learning

In my model an agent at position $x_T(t)$ moves at a constant speed, $\dot{x}_T(t) = v_T = 10$ cm s^{-1} from left to right around a periodic 1D track of circumference 2 m. A small reward density, $R(x)$, is centred at the far end of the track (Figure 2.1b). The goal of the agent is to learn the value function for the current policy, defined as the discounted integral of future reward

$$V^{\pi}(x) = \int_{t}^{\infty} e^{-\frac{t'-t}{\tau}} R(x(t'))dt' \;\mid\; x(t) = x \tag{B.1}$$

over a discount time horizon $\tau = 4$ s. This is done using a linear approximation, a weighted sum of independent features

$$\hat{V}^{\pi}(x) = \sum_{i} w_i \phi_i(x) \approx V^{\pi}(x). \tag{B.2}$$

Famously, this problem can be solved with a temporal difference learning rule

$$\dot{w}_i(t) = \eta\delta(t)z_i(t) \tag{B.3}$$

where $\delta(t)$ is the (temporally continuous) TD error

$$\delta(t) = R(t) + \frac{d\hat{V}^\pi(t)}{dt} - \frac{\hat{V}(t)}{\tau} \tag{B.4}$$

and $z_i(t)$ is the *eligibility trace* of the i^{th} feature

$$z_i(t) = \int_{-\infty}^{t} e^{\frac{t'-t}{\tau_z}}\phi_i(x(t'))dt'. \tag{B.5}$$

where $\tau_z \in [0, \tau]$ is the decay time scale of the eligibility trace. The basis features are a set of 200 small Gaussian receptive fields ($\sigma = 2$ cm, 95.45% firing field therefore measures $4\sigma = 8$ cm), roughly analogous to place cells (O'Keefe et al. 1971) in the hippocampal formation, evenly spaced at 1 cm intervals along the track. Their small size means each feature overlaps with approximately only its nearest neighbours. The reward density is another equally sized Gaussian at 1.95 m along the track. I choose this policy evaluation task because it admits an analytical solution for $V^\pi(x)$. Since the policy is non-stochastic one can evaluate the integral in eq. (B.1) accounting for the circular boundary conditions and compare this to the value estimate learnt by agents using temporal difference learning. All simulations (agent trajectory, theta sweeps, neural activities and policy evaluation) were produced using the RatInABox simulation package (George et al. 2024).

B.2.1 Relation to Discrete RL and TD(λ)

It is more common to see the temporally-*discrete* formulation of policy evaluation with TD learning (nb. for a full discussion/derivation of continuous

RL see Doya (2000)), summarised by

$$V^{\pi}(s_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} R(s_{t'}) \tag{B.6}$$

$$z_t = \sum_{t'=-\infty}^{t} (\lambda\gamma)^{t-t'} R(s_{t'}), \tag{B.7}$$

where t is now a discrete integer state index. γ, the 'discount factor' determines over how many future states the agent cares about reward and can be compared to τ, the temporally continuous 'discount time horizon', determining how long (as a unit of time) into the future the agent cares about reward. λ controls the decay-rate of the eligibility trace from $\lambda = 0$ (heavily bootstrapped regime) to $\lambda = 1$ (direct credit assignment, aka. online Monte Carlo). This discrete formulation is equivalent to the continuous one used here in the integral limit of short timesteps where the following relationships become apparent

$$\gamma = e^{-\frac{dt}{\tau}} \tag{B.8}$$

$$\gamma\lambda = e^{-\frac{dt}{\tau_z}}. \tag{B.9}$$

This enables a link to be made between the two extremes of TD(λ) as

$$TD(0) \Leftrightarrow \tau_z = 0 \tag{B.10}$$

$$TD(1) \Leftrightarrow \tau_z = \tau. \tag{B.11}$$

TD(0) (full-bootstrapping regime) occurs when the eligibility trace timescale falls to zero and TD(1) (Monte Carlo style learning) equates to when the eligibilty trace timescale matches the discount time horizon.

B.3 The Artificial Agent

The artificial agent learns according to the above TD learning rules and policy described in Appendix B.2 for a variety of eligibility trace timescales summarised in Table B.1: Note the inclusion of two extremes: TD(1) ($\tau_z = \tau = 4$) and

τ_z/s	4	2	1	0.5	0.25	0.125
τ/s	4	4	4	4	4	4
η_{opt}	0.4	0.5	0.6	0.8	1.1	1.3

Table B.1: Learning parameters for the artificial agents.

TD(∼0) ($\tau_z = 0.125 \approx 0$). In order to be sure that small learning rates were not bottlenecking learning I optimised η for each experiment by way of hyperparameter sweep (optimal value shown in table). When comparing the value estimate $\hat{V}^\pi(x)$ to the analytic value function $V^\pi(x)$ I use the coefficient of determination

$$R^2(V^\pi, \hat{V}^\pi) = 1 - \frac{\sum_x (\hat{V}^\pi(x) - V^\pi(x))^2}{\sum_x (V^\pi(x) - \langle V^\pi(x) \rangle_x)} \tag{B.12}$$

where the value estimate is first normalised to have the same maximum as $V^\pi(x)$ so, strictly, I am only comparing the shapes of the curves in Figure 2.1cd (bottom panels). The agent is allowed to explore and learn for a total of 640 s, exactly 32 laps, or until such a point that $R^2(V^\pi, \hat{V}^\pi)$ has been above 0.99 for the entire previous lap, whichever comes first. Agents start from a random initial position $x_T(0) \sim \mathsf{U}(0\text{ m}, 2\text{ m})$. Plots/error bars show the average/std over 50 such experiments in the case of the artificial agent and 10 in the case of the biological agent.

B.4 The Biological Agent

The biological agent differs from the artificial agent in two ways:

- **Short eligibility traces**: τ_z is fixed to a 0.01 s to emulate the biological constraint that neuronal memory times are short $\mathsf{O}(10)$ ms.

- **Theta sequences**: The firing rate of the features and the reward density are determined by the *encoded position* of the agent $x_E(t)$, not the true position $x_T(t)$. $x_E(t)$ sweeps from behind to in front of $x_T(t)$ in each theta cycle as described below.

Theta is modelled as a background oscillation of frequency $\nu_\theta = 1/T_\theta = 5$ Hz with a phase (used later) defined as

$$\phi_\theta(t) = \frac{t}{T_\theta} \mod 1 \tag{B.13}$$

During the middle fraction, $\beta = 0.75$ of each theta cycle $x_E(t)$ traverses symmetrically from behind to in front of the agent's true position at a speed of $v_E = v_T + v_S$ where v_S is the speed of $x_E(t)$ in the reference frame of the true position. Outside this window there is no registered position and all firing rates are zero. This can be stated formally as

$$x_E(t) = \begin{cases} x_T(t) + (\phi_\theta - 0.5)T_\theta v_S, & \text{if} \quad \frac{1-\beta}{2} < \phi_\theta(t) \leq \frac{1+\beta}{2} \\ \text{None}, & \text{otherwise} \end{cases} \tag{B.14}$$

which determines the neural firing rates used for learning $\phi_i(x_E(t))$ and $R(x_E(t))$ where $\phi_i(\text{None}) = R(\text{None}) := 0$. This leads to the core hypothesis of this chapter: **Since theta sequences traverse space faster than the real agent, the neural trajectory traverses the features faster than the real agent, compressing them. This compression means short eligibility traces, though remaining short, have more bang for their buck, effectively extending them**. The compression factor is

$$\kappa : \frac{v_E}{v_T} \implies \tau_z^{\text{eff}} = \kappa \tau_z. \tag{B.15}$$

Additionally, the same compression effect applies to the discount time horizon, τ, such that, in uncompressed time coordinates it will have effectively increased,

$$\tau^{\text{eff}} = \kappa \tau. \tag{B.16}$$

so in order to learn a value function with (effective) discount time horizon of $\tau = 4$ s, τ must be decreased accordingly. Table B.2 show the sweep velocities for six agents tested. These are carefully selected to match – according to

v_E/ms^{-1}	40	20	10	5	2.5	1.25
κ	400	200	100	50	25	12.5
τ_z/s	0.01	0.01	0.01	0.01	0.01	0.01
τ / s	0.01	0.02	0.04	0.08	0.16	0.32
η_{opt}	8	8	2	2	2	0.75
τ^{eff}	4	4	4	4	4	4
τ_z^{eff}/s	4	2	1	0.5	0.25	0.125

Table B.2: Learning parameters for the biological agents, Figure 2.1d and their artificial equivalents.

my theory – the eligibility trace timescales of the six artificial agents. Hence the final two rows show the 'effective' behaviour, i.e., if the theory is correct, which artificial agent (no theta sequences and any choice of τ_z) would this be equivalent to. Learning only occurs *within* sequences ($\frac{1-\beta}{2} < \phi_\theta(t) \leq \frac{1+\beta}{2}$). Outside this range (when there is no relevant data to learn from) learning is turned off ($\eta = 0$) reminiscent of the observation that hippocampal plasticity (LTP) oscillates significantly within each theta cycle (Hasselmo et al. 2014).

B.5 Analysis of Discontinuities in Theta Sequence Resets

The results shown in Figure 2.1d for the biological agent don't precisely converge to the value function for the slower sequences. I propose this may be due to 'loop-effects'. At the end of each theta cycle the sequence resets by discontinuously jumping back to a location behind the agent, Figure 2.1a. This discontinuity could induce errors to grow within the value estimate: whereas the neural activity *during* the sequence can be seen as a sped-up replica of the true state trajectory, this discontinuity does not reflect any real transition statistics. It is notable, therefore, that performance decay isn't catastrophic (all biological agents learn reasonable estimates of the value function) and is less pronounced for faster sequences, perhaps because the states at either end are further apart and interfere less. It is possible (but not tested) that the fraction of the cycle where there is no sweep ($1 - \beta$) allows existing short ETs to decay to zero

essentially "forgetting" the jump transition and ameliorating the problem.

Appendix C

Appendix to Chapter 3

C.1 Code Availability

Code to generate the results in Chapter 3 is available at `https://github.com/TomGeorge1234/STDP-SR`.

C.2 Spiking Neuron Model and STDP Learning Rule

The model comprises an agent exploring a maze where its position $\mathbf{x}$ at time t is encoded by the instantaneous firing of a population of N CA3 basis features, $f_j(\mathbf{x}, t)$ for $j \in \{1, \dots, N\}$. Each has a spatial receptive field given by a thresholded Gaussian of peak firing rate 5 Hz:

$$f_j^x\Big(\mathbf{x}(t)\Big) = \begin{cases} \text{Gaussian}\Big(\mathbf{x}_j, \sigma\Big) - c & \text{if } ||\mathbf{x}(t) - \mathbf{x}_j|| < 1\text{m} \\ 0 & \text{otherwise} \end{cases} \tag{C.1}$$

where $\mathbf{x}_j$ is the location of the field peak, $\sigma = 1$m is the standard deviation and c is a positive constant that keeps f_j^x continuous at the threshold.

The theta phase of the hippocampal local field potential oscillates at 10 Hz and is denoted by $\phi_\theta(t) \in [0, 2\pi]$. Phase precession suppresses the firing rate of a basis feature for all but a short period within each theta cycle. This period (and subsequently the time when spikes are produced, described in

more detail below) precesses earlier in each theta cycle as the agent crosses the spatial receptive field. Specifically, this is implemented by simply multiplying the spatial firing rate f_j^x by a theta modulation factor which rises and falls according to a von Mises distribution in each theta cycle, peaking at a 'preferred phase', ϕ_j^*, which depends on how far through the receptive field the agent has travelled (hence the spike timings implicitly encode location);

$$f_j^\theta\Big(\phi_\theta(t)\Big) = \text{VonMises}\Big(\phi_i^*, \kappa\Big) \tag{C.2}$$

where $\kappa = 1$ is the concentration parameter of the von Mises distribution. These basis features in turn drive a population of N downstream 'STDP successor features' (Equation (3.2)).

Firing rates of both populations ($f_j(\mathbf{x}, \phi_\theta)$ and $\tilde{\psi}_i(\mathbf{x}, \phi_\theta)$) are converted to spike trains according to an inhomogeneous Poisson process. These spikes drive learning in the synaptic weight matrix, W_{ij}, according to an STDP learning rule (details below). In summary, if a presynaptic CA3 basis feature fires immediately before a postsynaptic CA1 successor feature the binding strength between these cells is strengthened. Conversely, if they fire in the opposite order, their binding strength is weakened.

For comparison, successor feature learning is also implemented using a temporal difference (TD) learning rule, referred to as 'TD successor features', $\psi_i(\mathbf{x})$, to provide a ground truth against which the STDP successor features are compared. Like STDP successor features, these are constructed as a linear combination of basis features (Equation (3.3)).

Temporal difference learning updates M_{ij} as follows

$$\mathsf{M}_{ij} \leftarrow \mathsf{M}_{ij} + \eta\delta_{ij}^{\text{TD}} \tag{C.3}$$

where δ_{ij}^{TD} is the temporal difference error, which I derive below. In reinforcement learning the temporal difference error is used to learn discounted value functions (successor features can be considered a special type of value

function). It works by comparing an unbiased sample of the true value function to the currently held estimate. The difference between these is known as the temporal difference error and is used to update the value estimate until, eventually, it converges on (or close to) the true value function.

C.3 Phase Precession Details

In this hippocampal model CA3 place cells, referred to as basis features and indexed by j, have thresholded Gaussian receptive fields. The threshold radius is $\sigma = 1$ m and peak firing rate is $F = 5$ Hz. Mathematically, this is written as

$$f_j^x(\mathbf{x}(t)) = \frac{F}{1 - e^{-\frac{1}{2}}} \left[e^{-\frac{\|\mathbf{x}(t) - \mathbf{x}_j\|^2}{2\sigma^2}} - e^{-\frac{1}{2}} \right]_+, \tag{C.4}$$

where $[f(x)]_+ = max\Big(0, f(x)\Big)$, $\mathbf{x}_j$ is the centre of the receptive field and $\mathbf{x}(t)$ is the current location of the agent.

Phase precession is implemented by multiplying the spatial firing rate, $f_j^x(\mathbf{x})$, by a phase precession factor

$$f_j^\theta(\phi_\theta(t)) = 2\pi f_{\mathrm{VM}}\Big(\phi_\theta(t)\Big|\phi_j^*(\mathbf{x}), \kappa\Big). \tag{C.5}$$

where $f_{\mathrm{VM}}(x|\mu, \kappa)$ denotes the circular von Mises distribution on $x \in (0, 2\pi]$ with mean $\mu = \phi_j^*(\mathbf{x})$ and spread parameter $\kappa = 1$. This factor is large only when the current theta phase,

$$\phi_\theta(t) = 2\pi\nu_\theta t \pmod{2\pi}, \tag{C.6}$$

which oscillates at $\nu_\theta = 10$ Hz, is close to the cell's 'preferred' theta phase,

$$\phi_j^*(\mathbf{x}(t)) = \pi + \beta\pi d_j(\mathbf{x}(t)). \tag{C.7}$$

$d_j(\mathbf{x}(t)) \in [-1, 1]$ tracks how far through the cell's spatial receptive field, as

measured in units of σ, the agent has travelled:

$$d_j(\mathbf{x}(t)) = \frac{(\mathbf{x}(t) - \mathbf{x}_j) \cdot \frac{\dot{\mathbf{x}}(t)}{\|\dot{\mathbf{x}}(t)\|}}{\sigma}. \tag{C.8}$$

In instances where the agent travels directly across the centre of a cell (as is the case in 1D environments) then $(\mathbf{x}(t) - \mathbf{x}_j)$ and its normalised velocity (a vector of length 1, pointing in the direction of travel) $\frac{\dot{\mathbf{x}}(t)}{\|\dot{\mathbf{x}}(t)\|}$ are parallel such that $d_j(\mathbf{x})$ progresses smoothly in time from its minimum, -1, to its maximum, 1. In general, however, this extends to any arbitrary curved path an agent might take across the cell and matches the model used in Jeewajee et al. (2014). I fit β and κ to biological data in Fig. 5a of Jeewajee et al. (2014) Jeewajee et al. (2014) ($\beta = 0.5$, $\kappa = 1$). The factor of 2π normalises this term, although the instantaneous firing may briefly rise above the spatial firing rate $f_j^x(\mathbf{x})$, the average firing rate over the entire theta cycle is still given by the spatial factor $f_j^x(\mathbf{x})$. In total, the instantaneous firing rate of the basis feature is given by the product of the spatial and phase precession factors (Equation (3.1)).

Note that the firing rate of a cell depends explicitly on its location through the spatial receptive field (its "rate code") and implicitly on location through the phase precession factor (its "spike-time code") where location dependence is hidden inside the calculation of the preferred theta phase. Notably, the effect of phase precession is only visible on rapid "sub-theta" timescales. Its effect disappears when averaging over any timescale, T_{av} substantially longer than theta timescale of $T_\theta = 0.1$ s:

$$\frac{1}{T_{av}} \int_t^{t+T_{av}} f_j(\mathbf{x}(t), \phi_\theta(t'))dt' \approx \frac{1}{T_{av}} \int_t^{t+T_{av}} f_j^x(\mathbf{x}(t'))dt' \quad \text{for} \quad T_{av} >> T_\theta \tag{C.9}$$

This is important since it implies that the effect of phase precession is only important for synaptic processes with very short integration timescales, for example, STDP.

My phase precession model is "independent" (essentially identical to

Chadwick et al. (2015) Chadwick et al. (2015)) in the sense that each place cell phase precesses independently from what the other place cells are doing. In this model, phase precession directly leads to theta sweeps as shown in Figure 3.1. Another class of models referred to as "coordinated assembly" models Harris (2005) hypothesise that internal dynamics drive theta sweeps within each cycle because assemblies (aka place cells) dynamically excite one another in a temporal chain. In these models theta sweeps directly lead to phase precession. Feng and colleagues draw a distinction between theta precession and theta sequence, observing that while independent theta precession is evident right away in novel environments, longer and more stereotyped theta sequences develop over time Feng et al. (2015). Since I am considering the effect of theta precession on the formation of place field shape, the independent model is appropriate for this setting. I believe that considering how this model might relate to the formation of theta sequences or what implications theta sequences have for this model is an exciting direction for future work.

C.4 Synaptic Learning via STDP

STDP is a discrete learning rule: if a presynaptic neuron j fires before a postsynaptic neuron i their binding strength W_{ij}) is potentiated, conversely if the postsynaptic neuron fires before the presynaptic then weight is depressed. This is implemented as follows.

First, I convert the firing rates to spike trains. I sample, for each neuron, from an inhomogeneous spike train with rate parameter $f_j(\mathbf{x}, t)$ (for presynaptic basis features) or $\tilde{\psi}_i(\mathbf{x}, t)$ for postsynaptic successor features. This is done over the period $[0, T]$ across which the animal is exploring.

$$\Big(f_j(\mathbf{x}, t\Big), [0, T]) \overset{Poisson}{\longmapsto} \{t_j^{\text{pre}}\} \quad , \quad \Big(\tilde{\psi}_i(\mathbf{x}, t), [0, T]\Big) \overset{Poisson}{\longmapsto} \{t_i^{\text{post}}\} \quad \text{(C.10)}$$

Asymmetric Hebbian STDP is implemented online using a trace learning rule. Each presynaptic spike from CA3 cell, indexed j, increments an otherwise decaying memory trace, $T_j^{\text{pre}}(t)$, and likewise an analogous trace

for postsynaptic spikes from CA1, $T_i^{\text{post}}(t)$. I matched the STDP plasticity window decay times to experimental data: $\tau^{\text{pre}} = 20$ ms and $\tau^{\text{post}} = 40$ ms Bush et al. (2010).

$$\tau^{\text{pre}} \frac{dT_j^{\text{pre}}(t)}{dt} = -T_j^{\text{pre}}(t) + \sum_{t' \sim \{t_j^{\text{pre}}\}} \delta(t - t') \tag{C.11}$$

$$\tau^{\text{post}} \frac{dT_i^{\text{post}}(t)}{dt} = -T_i^{\text{post}}(t) + \sum_{t' \sim \{t_i^{\text{post}}\}} \delta(t - t'). \tag{C.12}$$

The model is simplified by fixing weights during learning:

$$\tilde{\psi}_i(\mathbf{x}, t) = \sum_j \mathsf{W}_{ij}^{\mathsf{A}} f_j(\mathbf{x}, t) \qquad \text{During learning} \tag{C.13}$$

where $\mathsf{W}_{ij}^{\mathsf{A}}$ will be referred to as the "anchoring" weights which, up until now, have been set to the identity $\mathsf{W}_{ij}^{\mathsf{A}} = \delta_{ij}$. Since $f_j(\mathbf{x}, t)$ is the *phase precessing* features, $\tilde{\psi}_i(\mathbf{x}, t)$ also inherits phase precession from these features mapped through $\mathsf{W}_{ij}^{\mathsf{A}}$. Fixing the weights means that during learning the effect of changes in W_{ij} are *not* propagated to the successor features (CA1), their influence is only considered during post-learning recall broadly analogous to the distinct encoding and retrieval phases that have been hypothesised to underpin hippocampal function Hasselmo et al. (2002). I relax this assumption in Figure C.2 and allow W_{ij} to be updated online, showing this isn't essential.

After a period, $[0, T]$ of exploration the synaptic weights are updated on aggregate to account for STDP.

$$\begin{aligned} \mathsf{W}_{ij}(T) = \mathsf{W}_{ij}(0) + \eta \Big[& \underbrace{a^{\text{pre}} \sum_{t_i \sim \{t_i^{\text{post}}\}} \delta(t - t_i) T_j^{\text{pre}}(t)}_{\text{“pre-before-post potentiations”}} \\ & + \underbrace{a^{\text{post}} \sum_{t_j \sim \{t_j^{\text{pre}}\}} \delta(t - t_j) T_i^{\text{post}}(t)}_{\text{“post-before-pre depressions”}} \Big] \end{aligned} \tag{C.14}$$

where the second terms accounts for the cumulative potentiation and depression

due to STDP from spikes in the CA3 and CA1 populations. η is the learning rate (here set to 0.01) and a^{pre} and a^{post} give the relative amounts of pre-before-post potentiation and post-before-pre depression, set to match experimental data from Bi et al. (1998) as 1 and -0.4 respectively. The weights are initialised to the identity: $\mathsf{W}_{ij}(0) = \delta_{ij}$.

Finally, when analysing the successor features after learning I use the updated weight matrix, not the anchoring weights, (and turn off phase precession since I am only interested in rate maps)

$$\tilde{\psi}_i(\mathbf{x}) = \sum_j \mathsf{W}_{ij}(T) f_j^x(\mathbf{x}). \qquad \text{After learning} \tag{C.15}$$

C.5 Temporal Difference Learning

To test the hypothesis that STDP is a good approximation to TD learning, the *TD successor features* were simultaneously computed, defined as the total expected future firing of a basis feature:

$$\psi_i(\mathbf{x}) = \mathbb{E}\left[\int_t^\infty \frac{1}{\tau} e^{-\frac{t'-t}{\tau}} f_i^x\Big(\mathbf{x}(t')\Big) dt' \;\Big|\; \mathbf{x}(t) = \mathbf{x} \right]. \tag{C.16}$$

τ is the temporal discounting time-horizon (related to γ, the discount factor used in reinforcement learning on temporally discretised MDPs, $\gamma = e^{-\frac{dt}{\tau}}$) and the expectation is over trajectories initiated at position $\mathbf{x}$. This formula explains the one-to-one correspondence between CA3 cells and CA1 cells in my hippocampal model (Figure 3.1b): each CA1 cell, indexed i, learns to approximate the TD successor feature for its target basis feature, also indexed i. I set the discount timescale to $\tau = 4$ s to match relevant behavioural timescales for an animal exploring a small maze environment where behavioural decisions, such as whether to turn left or right, need to be made with respect to optimising future rewards occurring on the order of seconds.

I learn these successor features by tuning the weights of a linear

decomposition over the basis feature set:

$$\psi_i(\mathbf{x}) = \sum_j \mathsf{M}_{ij} f_j^x(\mathbf{x}), \tag{C.17}$$

this way M_{ij} can be directly compared to the STDP weight matrix W_{ij}.

My TD successor matrix, M_{ij}, should not be confused with the successor *representation* as defined in Stachenfeld et al. Stachenfeld et al. (2017) and denoted $M(\mathsf{s}_i, \mathsf{s}_j)$, though they are analogous. M_{ij} can be thought of as an analogue to $M(\mathsf{s}_i, \mathsf{s}_j)$ for spatially continuous (i.e. not one-hot) basis features, I show in Appendix C.6.1 that they are equal (strictly, $M(\mathsf{s}, \mathsf{s}') = \mathsf{M}_{ij}^{\mathsf{T}}$) in the limit of a discrete one-hot place cells.

Temporal difference learning The temporal difference (TD) update rule is used to learn the TD successor matrix (Equation (C.17)). The standard TD(0) learning rule for a linear value function, $\psi_i(\mathbf{x})$, which basis feature weights M_{ij} is Sutton et al. (1998):

$$\mathsf{M}_{ij} \leftarrow \mathsf{M}_{ij} + \eta \delta_i f_j^x(\mathbf{x}) \tag{C.18}$$

where δ_i is the observed TD-error for the i^{th} successor feature and η is the learning rate. Note that I am only considering the spatial component of the firing rate, $f_j^x(\mathbf{x})$, not the phase modulation component, $f_j^\theta(\mathbf{x})$, which (as shown) would average away over any timescale significantly longer than the theta timescale (100 ms). For now I will drop the superscript and write $f_j^x(\mathbf{x}) = f_j(\mathbf{x})$

To find the TD-error I must derive a temporally continuous analogue of the Bellman equation. Following Doya (2000) I take the derivative of Equation (C.16) which gives a consistency equation on the successor feature as

follows:

$$\frac{d}{dt}\psi_i\Big(\mathbf{x}(t)\Big) = \frac{d}{dt}\int_t^\infty \frac{1}{\tau}e^{-\frac{t'-t}{\tau}} f_i\Big(\mathbf{x}(t')\Big)dt' \tag{C.19}$$

$$= \frac{1}{\tau}\Bigg(\psi_i\Big(\mathbf{x}(t)\Big) - f_i\Big(\mathbf{x}(t)\Big)\Bigg) \tag{C.20}$$

This gives a continuous TD-error of the form

$$\delta_i(t) = \frac{d}{dt}\psi_i\Big(\mathbf{x}(t)\Big) + \frac{1}{\tau}\Bigg(f_i\Big(\mathbf{x}(t)\Big) - \psi_i\Big(\mathbf{x}(t)\Big)\Bigg) \tag{C.21}$$

which can be rediscretised and rewritten by Taylor expanding the derivative ($\psi_i(t) = \frac{\psi_i(t)-\psi_i(t-dt)}{dt}$) to give

$$\delta_i(t) = \frac{1}{dt}\Bigg(\frac{dt}{\tau}f_i\Big(\mathbf{x}(t)\Big) + \Big(1-\frac{dt}{\tau}\Big)\psi_i\Big(\mathbf{x}(t)\Big) - \psi_i\Big(\mathbf{x}(t-dt)\Big)\Bigg). \tag{C.22}$$

This looks like a conventional TD-error term (typically something like $\delta_t = \mathsf{R}_t + \gamma\mathsf{V}_t - \mathsf{V}_{t-1}$) except that we can choose dt (the timestep between learning updates) freely. Finally expanding $\psi_i(\mathbf{x}(t))$ using (Equation (3.3)) and substituting this back into Equation (C.18) gives the update rule:

$$\mathsf{M}_{ij} \leftarrow \mathsf{M}_{ij} + \frac{\eta}{dt}\Bigg[\frac{dt}{\tau}f_i\Big(\mathbf{x}(t)\Big) + \sum_k \mathsf{M}_{ik}\Big[\Big(1-\frac{dt}{\tau}\Big)f_k\Big(\mathbf{x}(t)\Big) - f_k\Big(\mathbf{x}(t-dt)\Big)\Big]\Bigg] f_j\Big(\mathbf{x}(t)\Big). \tag{C.23}$$

This rule doesn't stipulate a fixed time step between updates. Unlike traditional TD updates rules on discrete MDPs, dt can take *any* positive value. The ability to adaptively vary dt has potentially underexplored applications for efficient learning: when information density is high (e.g. when exploring new or complex environments, or during a compressed replay event Skaggs et al. (1996a)) it may be desirable to learn regularly by setting dt small. Conversely when the information density is low (for example in well known or simple environments) or learning is undesirable (for example the agent is aware that

a change to the environment is transient and should not be committed to memory), dt can be increased to slow learning and save energy. In practice, the agent is set to perform a learning update approximately every 1 cm along its trajectory ($dt \approx 0.1$ s).

I add a small amount of $\mathsf{L2}$ regularisation by adding the term $-2\eta\lambda\mathsf{M}$ to the right hand side of Equation (C.24). This breaks the degeneracy in M_{ij} caused by having a set of basis features which is overly rich to construct the successor features and can be interpreted, roughly, as a mild energy constraint favouring smaller synaptic connectomes. In total the full update rule from our TD successor matrix in matrix form is given by

$$\mathsf{M} \leftarrow \mathsf{M} + \frac{\eta}{dt}\left[\frac{dt}{\tau}\mathbf{f}\Big(\mathbf{x}(t)\Big) + \mathsf{M}\left[\Big(1-\frac{dt}{\tau}\Big)\mathbf{f}\Big(\mathbf{x}(t)\Big) - \mathbf{f}\Big(\mathbf{x}(t-dt)\Big)\right]\right]\mathbf{f}^{\mathsf{T}}\Big(\mathbf{x}(t)\Big) - 2\eta\lambda\mathsf{M}. \tag{C.24}$$

C.6 Continuous Successor Features

Typically, as in Stachenfeld et al. Stachenfeld et al. (2017), the successor *representation* is calculated in discretised time and space. $M(\mathsf{s}_i, \mathsf{s}_j)$ encodes the expected discounted future occupancy of state s_j along a trajectory initiated in state s_i:

$$M(\mathsf{s}_i, \mathsf{s}_j) = \mathbb{E}\left[\sum_{t=0}\gamma^t\delta(\mathsf{s}_t = \mathsf{s}_j) \;\middle|\; \mathsf{s}_0 = \mathsf{s}_i\right] \tag{C.25}$$

There are two forms of discretisation here. Firstly, time is discretised: it increases by a fixed increment, $+1$, to transition the state from $\mathsf{s}_t \rightarrow \mathsf{s}_{t+1}$. Secondly, assuming this is a spatial exploration task, space is discretised: the agent can be in exactly one state on any given time.

Both these constraints are loosened, reinstating time and space as continuous quantities. Since, for space, I cannot hope to enumerate an infinite number of locations, I represent the state by a population vector of diffuse, overlapping spatially localised place cells. Thus it is no longer meaningful to

ask what the expected future occupancy of a single location will be. The closest analogue, since the place cells are spatially localised, is to ask how much I expect place cell, i, centred at $\mathbf{x}_i$, to fire in the near (discounted) future. This continuous time constraint alters the sum over time into an integral over time. Further, the role of γ which discounts state occupancy many time steps into the future, is replaced by τ which discounts firing a long time into the future. Thus the extension of the successor representation, $M(\mathsf{s}_i, \mathsf{s}_j)$, to continuous time and space is given by the successor *feature*,

$$\psi_i(\mathbf{x}) = \mathbb{E}\left[\int_t^\infty \frac{1}{\tau} e^{-\frac{t'-t}{\tau}} f_i\big(\mathbf{x}(t')\big) dt' \;\middle|\; \mathbf{x}(t) = \mathbf{x}\right]. \tag{C.26}$$

Why have I chosen to do this? Temporally it makes little sense to discretise time in a continuous exploration task: γ, the reinforcement learning discount factor, describes how many timesteps into the future the predictive encoding accounts for and so undesirably ties the predictive encoding to the otherwise arbitrary size of the simulation timestep, dt. In the continuous definition, τ intuitively describes how long into the future the predictive encoding discounts over and is independent of dt. This definition allows for online flexibility in the size of dt, as shown in Equation (C.24). This relieves the agent of a burden imposed by discretisation; namely that it must learn with a fixed time step, +1, all the time. Now the agent potentially has the ability to choose the fidelity over which to learn and this may come with significant benefits in terms of energy efficiency, as described above. Further, using the discretised form implicitly ties the definition of the successor representation (or any similarly defined value function) to the time step used in their simulation.

When space *is* discretised, the successor representation is a matrix encoding predictive relationships between these discrete locations. TD successor features, defined above, are the natural extension of the successor representation in a continuous space where location is encoded by a population of overlapping basis features, rather than exclusive one-hot states. The TD successor matrix, M_{ij},

can most easily be viewed as set of driving weights: M_{ij} is large if basis feature $f_j(\mathbf{x})$ contributes strongly to successor feature $\psi_i(\mathbf{x})$. They are closely related (for example, in the effectively discrete case of non-overlapping basis features, it can be shown that the TD successor matrix then corresponds directly to the transpose of the successor representation, $\mathsf{M}_{ij}^{\mathsf{T}} = \mathsf{M}(\mathsf{s}_i, \mathsf{s}_i)$, see below for proof) but I believe the continuous case has more applications in terms of biological plausibility; electrophysiological studies show hippocampus encodes position using a population vector of overlapping place cells, rather than one-hot states. Furthermore the continuous case maps neatly onto known neural circuitry, as in my case with CA3 place cells as basis features, CA1 place cells as successor features, and the successor matrix as the synaptic weights between them. In this case, the choice not to discretise space and use a more biologically compatible basis set of large overlapping place cells is necessary - were the basis features to not overlap they would not be able to reliably form associations using STDP since often only one cell would ever fire in a given theta cycle.

For completeness (though this is not something studied in this report) this continuous successor feature form also allows for rapid estimation of the value function in a neurally plausible way. Whereas for the discrete case value can be calculated as:

$$V(\mathsf{s}_i) = \sum_j M(\mathsf{s}_i, \mathsf{s}_j) R(\mathsf{s}_j) \tag{C.27}$$

where $R(\mathsf{s}_j)$ is the per-time-step reward to be found at state s_j, for continuous successor feature setting:

$$\mathsf{V}(\mathbf{x}) = \sum_j \psi_j(\mathbf{x}) \mathsf{R}_j \tag{C.28}$$

where R_j is a vector of weights satisfying $\sum_j \mathsf{R}_j f_j(\mathbf{x}) = R(x)$ where $R(x)$ is the reward-rate found at location $\mathbf{x}$. Equation (C.28) can be confirmed by substituting into it Equation (C.26). R_j (like $R(\mathsf{s}_j)$) must be learned independent to, and as well as, the successor features, a process that is not the

focus of this study although correlates have been observed in the hippocampus Gauthier et al. (2018). $\mathsf{V}(\mathbf{x})$ is the temporally continuous value associated with trajectories initialised at $\mathbf{x}$:

$$\mathsf{V}(\mathbf{x}) = \mathbb{E}\left[\int_t^\infty \frac{1}{\tau} e^{-\frac{t'-t}{\tau}} R\big(\mathbf{x}(t')\big) dt' \;\middle|\; \mathbf{x}(t) = \mathbf{x}\right]. \tag{C.29}$$

C.6.1 Equivalence of the TD successor matrix to the successor representation

Here I show the equivalence between $M(\mathsf{s}_i, \mathsf{s}_j)$ and M_{ij}. First I can rediscretise time by setting dt' to be constant and defining $\gamma = 1 - \frac{dt'}{\tau}$ and $\mathbf{x}_n = \mathbf{x}(n \cdot dt')$. The integral in Equation (C.26) becomes a sum,

$$\psi_i(\mathbf{x}) = (1-\gamma)\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t f_i\big(\mathbf{x}_t\big) \;\middle|\; \mathbf{x}_0 = \mathbf{x}\right]. \tag{C.30}$$

Next I rediscretise space by supposing that CA3 place cells in my model have strictly non-overlapping receptive fields which tile the environment. For each place cell, i, there is continuous area, $\mathcal{A}_i$, such that for any location within this area place cell i fires at a constant rate whilst all others are silent. When $\mathbf{x} \in \mathcal{A}_i$ we denote this state $\mathsf{s}(\mathbf{x}) = \mathsf{s}_i$ (since all locations in this area have identical population vectors).

$$f_i(\mathbf{x}) = \delta(\mathbf{x} \in \mathcal{A}_i) = \delta\big(\mathsf{s}(\mathbf{x}) = \mathsf{s}_i\big) \tag{C.31}$$

Let the initial state be $\mathsf{s}(\mathbf{x}) = \mathsf{s}_j$ (i.e. $\mathbf{x} \in \mathcal{A}_j$). Putting this into Equation (C.30) and equating to Equation (3.3), the definition of our TD successor matrix, gives

$$\psi_i(\mathbf{x}) = \sum_k \mathsf{M}_{ik}\delta(\mathsf{s}_j = \mathsf{s}_k) = (1-\gamma)\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \delta(\mathsf{s}_t = \mathsf{s}_i) \;\middle|\; \mathsf{s}_0 = \mathsf{s}_j\right], \tag{C.32}$$

confirming that

$$\mathsf{M}^{\mathsf{T}}_{ij} \propto M(\mathsf{s}_i, \mathsf{s}_j). \tag{C.33}$$

C.7 Relation to RatInABox

Note to reader: this work chronologically predates the work presented in Chapter 1 and so does not *officially* use the RatInABox toolkit. However, the motion and cell models developed in this work became an early prototype for the toolkit and many of the simulation details are similar.

C.8 Simulation Details

Maze details In the 1D open loop maze (Figure 3.2a-e) the policy was to always move around the maze in one direction (left to right, as shown) at a constant velocity of 16 cm s^{-1} along the centre of the track. Although figures display this maze as a long corridor it is topologically identical to a loop; place cells close to the left or right sides have receptive fields extending into the right or left of the corridor respectively. 50 Gaussian basis features of radius 1 m, as described above, are placed with their centres uniformly spread along the track. Agents explored for a total time of 30 minutes.

In the 1D corridor maze, Figure 3.2f-j, the situation is only changed in one way: the left and right hand edges of the maze are closed by walls. When the agent reaches the wall it turns around and starts walking the other way until it collides with the other wall. Agents explored for a total time of 30 minutes.

In the 2D two room maze, 200 basis feature are positioned in a grid across the two rooms (100 per room) then their location jittered slightly (Figure 3.2k). The cells are geodesic Gaussians. This means that the $\|\mathbf{x}(t) - \mathbf{x}_i\|^2$ term in Equation (C.4) measures the distance from the agent location the centre of cell i along the shortest walk that complies with the wall geometry. This explains the bleeding of the basis feature through the door in Figure 3.3f. Agents explored for a total time of 120 minutes.

The movement policy of the agent is a random walk with momentum. The agent moves forward with the speed at each discrete time step drawn from a Rayleigh distribution centred at 16 cm s^{-1}. At each time step the agent rotates a small amount; the rotational speed is drawn from a normal distribution centred at zero with standard deviation 3π rad s^{-1} (π rad s^{-1} for the 1D mazes). Whenever the agent gets close to a wall (within 10 cm) the direction of motion is changed parallel to the wall, thus biasing towards trajectories that "follow" the boundaries, as observed in real rats. This model was designed to match closely the behaviour of freely exploring rats and was adapted from the model initially presented in Raudies and Hasselmo, 2012 Raudies et al. (2012). I add one additional behavioural bias: in the 2D two room maze, whenever the agent passes within 1 metre of the centre point of the doorway connecting the two rooms its rotational velocity is biased to turn it towards the door centre. This has the effect of encouraging room-to-room transitions, as is observed in freely moving rats Carpenter et al. (2015).

Analyses of the STDP and TD successor matrices For the 1D mazes there exists a translational symmetry relating the $N = 50$ uniformly distributed basis features and their corresponding rows in the STDP/TD weight matrices. This symmetry is exact for the 1D loop maze (all cells around a circle are rotated versions of one another) and approximate for the corridor maze (broken only for cells near to the left or right bounding wall). The result is that much of the information in the linear track weight matrices Figure 3.2b,c,g,h can be viewed more easily by collapsing this matrix over the rows centred on the diagonal entry (plotted in Figure 3.2d and i). This is done using a circular permutation of each matrix row by a count, n_i, equal to how many times we must shift cell i to the right in order for its centre to lie at the middle of the track, $x_i = 2.5$m,

$$\mathsf{W}_{ij}^{\text{aligned}} = \mathsf{W}_{i,(j+n_i \pmod{50})}. \tag{C.34}$$

This is the 'row aligned matrix'. Averaging over its rows removes little information thanks to the symmetry of the circular track. I therefore define the 1D quantity

$$\langle \mathsf{W} \rangle_j := \frac{1}{N} \sum_{i=1}^{N} \mathsf{W}_{ij}^{\text{aligned}}. \tag{C.35}$$

which is a convenient way to plot, in 1D, only the non-redundant information in the weight matrices.

C.9 A Theoretical Connection Between STDP and TD Learning

Why does STDP between phase precessing place cells approximate TD learning? This section attempts to shed some light on this question by analytically studying the equations of TD learning. Ultimately, comparisons between these learning rules are difficult since the former is inherently a discrete learning rule acting on pairs of spikes whereas the latter is a continuous learning rule acting on firing rates. Nonetheless, in the end the following conclusions are drawn:

1. In the first part it will be shown that, under a small set of biologically feasible assumptions, temporal difference learning "looks like" a spike-time dependent temporally-asymmetric Hebbian learning rule (that is, roughly, STDP) where the temporal discount time horizon, τ is equal to the synaptic plasticity timescale $\mathsf{O}(20 \text{ ms})$.

2. In the second part it will be seen that this limitation that the temporal discount time horizon is restricted to the timescale of synaptic plasticity (i.e. very short) can be overcome by compressing the inputs. Phase precession, or more formally, theta sweeps, perform exactly the required compression.

In sum, there is a deep connection between TD learning and STDP and the role of phase precession is to compress the inputs such that a very

short predictive time horizon amounts to a long predictive time horizon in decompressed time coordinates. This section will finish by discussing where these learning rules diverge and the consequences of their differences on the learned representations. The goal here is not to derive a mathematically rigorous link between STDP and TD learning but to show that a connection exists between them and to point the reader to further resources if they wish to learn more.

C.9.1 Reformulating TD learning to look like STDP

First, recall that the temporal difference (TD) rule for learning the successor features $\psi_i(\mathbf{x})$ defined in Equation (C.16) takes the form:

$$\frac{d\mathsf{M}_{ij}}{dt} = \eta\delta_i(t)\mathsf{e}_j(t) \tag{C.36}$$

where M_{ij} are the weights of the linear function approximator, Equation (3.3)[1] and $\delta_i(t)$ is the continuous temporal difference error defined in Equation (C.21). $\mathsf{e}_j(t)$ is the *eligibility trace* for feature j defined according to

$$\mathsf{e}_j(t) = \int_{-\infty}^{t} \frac{1}{\tau_\mathsf{e}} e^{\frac{t-t'}{\tau_\mathsf{e}}} f_j(\mathbf{x}(t'))dt' \tag{C.37}$$

or, equivalently, by its dynamics (which I will make use of)

$$\mathsf{e}_j(t) = f_j(t) - \tau_\mathsf{e}\dot{\mathsf{e}}_j(t). \tag{C.38}$$

where $\tau_\mathsf{e} \in [0, \tau]$ is a 'free' parameter, the eligibility trace timescale, analogous to λ in discrete TD(λ). When $\tau_\mathsf{e} = 0$, the learning rule used to learn successor features, "TD(0)", is recovered, as in Equation (C.18).

Subbing Equation (C.21) and Equation (C.38) into this update rule,

[1]Note, firstly, it is a coincidence specific to this study that the basis features of the linear function approximator, Equation (3.3), happen to be the same features of which we are computing the successor features, Equation (C.16). In general this needn't be the case. Secondly, this analysis applies to *any* value function, not just successor features which are a specific example. If $f_i(\mathbf{x})$ in Equation (C.16) was a reward density then $\psi_i(\mathbf{x})$ would become a true value function (discounted sum of future rewards) in the more conventional sense.

Equation (C.36), rearranges to give

$$\frac{d\mathsf{M}_{ij}}{dt} = \eta\Big(f_i\mathsf{e}_j - \psi_i f_j + \tau\dot{\psi}_i\mathsf{e}_j - \tau_\mathsf{e}\psi_i\dot{\mathsf{e}}_j\Big) \tag{C.39}$$

where η is redefined as $\eta \leftarrow \eta' = \eta/\tau$. Now let the predictive time horizon be equal to the eligibility trace timescale. This setting is also called TD(1) or Monte Carlo learning,

$$\tau = \tau_\mathsf{e} \tag{C.40}$$

Now

$$\frac{d\mathsf{M}_{ij}}{dt} = \eta\Big(f_i\mathsf{e}_j - \psi_i f_j + \tau_\mathsf{e}\frac{d}{dt}(\psi_i\mathsf{e}_j)\Big). \tag{C.41}$$

The final term in this update rule, the total derivative, can be ignored with respect to the stationary point of the learning process. To see why, consider the simple case of a periodic environment which repeats over a time period T – this is true for the 1D experiments studied here. Learning is at a stationary point when the integrated changes in the weights vanish over one whole period:

$$0 = \int_t^{t+T} dt'\dot{\mathsf{M}}_{ij}(t') = \eta\int_t^{t+T} dt'\Big(f_i\mathsf{e}_j - \psi_i f_j\Big) + \eta\tau_\mathsf{e}\int_t^{t+T} dt'\frac{d}{dt'}(\psi_i(t')\mathsf{e}_j(t')) \tag{C.42}$$

$$= \eta\int_t^{t+T} dt'\Big(f_i\mathsf{e}_j - \psi_i f_j\Big) + \eta\tau_\mathsf{e}\Big[\psi_i(t+T)\mathsf{e}_j(t+T) - \psi_i(t)\mathsf{e}_j(t)\Big] \tag{C.43}$$

$$= \eta\int_t^{t+T} dt'\Big(f_i\mathsf{e}_j - \psi_i f_j\Big) \tag{C.44}$$

where the last term vanishes due to the periodicity. This shows that the learning rule converges to the same fixed point (i.e. the successor feature) irrespective of whether this term is present and it can therefore be removed. The dynamics of this updated learning rule won't strictly follow the same trajectory as TD learning but they will converge to the same point. Although strictly I only showed this to be true in the artificially simple setting of a periodic environment

it is more generally true in a stochastic environment where the feature inputs depend on a stationary latent Markov chain Brea et al. (2016).

Thus a valid learning rule which converges onto the successor feature can be written as

$$\frac{d\mathsf{M}_{ij}}{dt} = \eta\Big(f_i(t)\mathsf{e}_j(t) - \psi_i(t)f_j(t)\Big) \tag{C.45}$$

Claim: this looks like a continuous analog of STDP acting on the weights between a set of input features, indexed j, and a set of downstream "successor features" indexed i. Each term in the above learning rule can be non-rigorously identified as follows, a key change is that the successor features neurons have two-compartments; a somatic compartment and a dendritic compartment:

- $f_i(t) := \mathsf{V}_i^{\text{soma}}(t)$ is the somatic membrane voltage which is primarily set by a "target signal". In general this target signal could be any reward density function, here it is the firing rate of the i$^{\text{th}}$ input feature.
- $\psi_i(t) := \mathsf{V}_i^{\text{dend}}(t)$ is the voltage inside a dendritic compartment which is a weighted linear sum of the input currents, Equation (3.3). This compartment is responsible for learning the successor feature by adjusting its input weights, M_{ij}, according to equation (Equation (C.45)).
- $f_j(t) := \mathsf{I}_j(t)$ are the synaptic currents into the dendritic compartment from the upstream features.
- $\mathsf{e}_j(t) := \tilde{\mathsf{I}}_j(t)$ are the low-pass filtered eligibility traces of the synaptic input currents.

$$\frac{d\mathsf{M}_{ij}}{dt} = \eta\Big(\underbrace{\mathsf{V}_i^{\text{soma}}(t)\tilde{\mathsf{I}}_j(t)}_{\text{pre-before-post potentiation}} - \underbrace{\mathsf{V}_i^{\text{dend}}(t)\mathsf{I}_j(t)}_{\text{post-before-pre depression}}\Big) \tag{C.46}$$

This learning rule, mapped onto the synaptic inputs and voltages of a two-compartment neuron, is Hebbian. The first term potentiates the synapse M_{ij} if there is a correlation between the low-pass filtered presynaptic current and the somatic voltage (which drives postsynaptic activity). More specifically this potentiation is is temporally asymmetric due to the second term which sets a

threshold. A postsynaptic spike (e.g. when $\mathsf{V}_i^{\text{soma}}(t)$ reaches threshold) will cause potentiation if

$$\mathsf{V}_i^{\text{soma}}(t)\tilde{\mathsf{I}}_j(t) > \mathsf{V}_i^{\text{dend}}(t)\mathsf{I}_j(t) \tag{C.47}$$

but since the eligibility trace decays uniformly after a presynaptic input this will only be true if the postsynaptic spike arrives very soon after. This is *pre-before-post* potentiation. Conversely an unpaired presynaptic input (e.g. when $\mathsf{I}_j(t)$ spikes) will likely cause depression since this bolsters the second depressive term of the learning rule but not the first (note this is true if its synaptic weight is positive such that $\mathsf{V}^{\text{dend}}(t)$ will be high too). This is analogous to *post-before-pre* depression. Whilst not identical, it is clear this rule bears the key hallmarks of the STDP learning rule used in this study, specifically: pre-before-post synaptic activity potentiates a synapse if post synaptic activity arrive within a short time of the presynaptic activity and, secondly, post-before-pre synaptic activity will typically result in depression of the synapse.

Intuitively it now makes sense why asymmetric STDP learns successor features. If a postsynaptic spike from the i^{th} neuron arrives just after a presynaptic spike from the j^{th} feature it means, in all probability, that the presynaptic input features is "predictive" of whatever caused the postsynaptic spike which in this case is the i^{th} feature. Thus if we want to learn a function which is predictive of the i^{th} features future activity (its successor feature) we should increase the synaptic weight M_{ij}. Finally, identifying that this learning rule looks similar to STDP fixes the timescale of the eligibility trace to be the timescale of STDP plasticity i.e. $\mathsf{O}(20-50\text{ ms})$. And to derive this learning rule we required that the temporal discount time horizon must equal the eligibility trace timescale, altogether:

$$\tau = \tau_{\mathsf{e}} = \tau_{\text{STDP}} \approx 20 - 50\text{ ms} \tag{C.48}$$

This limits the predictive time horizon of the learnt successor feature to a rather useless – but importantly non-zero – 20-50 ms. In the next section I will show how phase precession presents a novel solution to this problem.

C.9.2 Theta phase precession compresses the temporal structure of input features

It was shown in Figure 3.1 how phase precession leads to theta sweeps. These phenomena are two sides of the same coin. This section will start by positing the existence of theta sweeps and showing that this leads to a potentially large amount of compression of the feature basis set in time.

First, consider two different definitions of position. $\mathbf{x}_T(t)$ is the "True" position of the agent representing where it is in the environment at time t. $\mathbf{x}_E(t)$ is the "Encoded" position of the agent which determines the firing rate of place cells which have spatial receptive fields $f_i(\mathbf{x}_E(t))$. During a theta sweep the encoded position $\mathbf{x}_E(t)$ moves with respect to the true position $\mathbf{x}_T(t)$ at a relative speed of $\mathbf{v}_S(t)$ where the subscript S distinguishes the "Sweep" speed from the absolute speed of the agent $\dot{\mathbf{x}}_T(t) = \mathbf{v}_A(t)$. In total, accounting for the motion of the agent:

$$\dot{\mathbf{x}}_E(t) = \mathbf{v}_A(t) + \mathbf{v}_S(t) \tag{C.49}$$

Now consider how the population activity vector changes in time

$$\frac{d}{dt} f_i^T(\mathbf{x}_E(t)) = \nabla_{\mathbf{x}} f_i^T(\mathbf{x}) \cdot \dot{\mathbf{x}}_E(t) = \nabla_{\mathbf{x}} f_i^T(\mathbf{x}) \cdot (\mathbf{v}_A(t) + \mathbf{v}_S(t)) \tag{C.50}$$

and compare the time how it would varying in time if there was no theta sweep (i.e $\mathbf{x}_E(t) = \mathbf{x}_T(t)$)

$$\frac{df_i^T(\mathbf{x}_T(t))}{dt} = \nabla_{\mathbf{x}} f_i^T(\mathbf{x}) \cdot \frac{d\mathbf{x}_T(t)}{dt} = \nabla_{\mathbf{x}} f_i^T(\mathbf{x}) \cdot \mathbf{v}_A(t). \tag{C.51}$$

They are proportional. Specifically in 1D, where the sweep is observed to move in the same direction as the agent (from behind it to in front of it) this amounts

to compression of the temporal dynamics by a factor of

$$k_\theta = \frac{v_A + v_S}{v_A}. \tag{C.52}$$

This "compression" is also true in 2D where sweeps are also observed to move largely in the same direction as the agent.

If this compression is large it would solve the timescale problem described above. This is because learning a successor feature with a very small time horizon, τ, where the input trajectory is heavily compressed in time by a factor of κ_θ amounts *to the same thing* as learning a successor feature with a long time horizon $\tau' = \tau\kappa_\theta$ where the inputs are not compressed in time.

What is v_S, and is it fast enough to provide enough compression to learn temporally extended SRs? I can make a very rough ballpark estimate. Data is hard to come by but studies suggest the intrinsic speed of theta sweeps can be quite fast. Figures in Feng et al. (2015), Wang et al. (2020) and Bush et al. (2022) show sweeps moving at up to, respectively, 9.4 ms^{-1}, 8.5 ms^{-1} and 2.3 ms^{-1}. A conservative range estimate of $v_S \approx 5 \pm 5$ ms^{-1} accounts for very fast and very slow sweeps. The timescale of STDP is debated but a reasonable conservative estimate would be around $\tau_{\text{STDP}} \approx 35 \pm 15 \times 10^{-3}$ s which would cover the range of STDP timescales I use here. The typical speed of a rat, though highly variable, is somewhere in the range $v_A \approx 0.15 \pm 0.15$ ms^{-1}. Combining these (with correct error analysis, assuming Gaussian uncertainties) gives an effective timescale increase of

$$\tau' = \tau k_\theta = \tau_{\text{STDP}} \frac{v_A + v_S}{v_A} \approx 1.1 \pm 1.7\text{s} \tag{C.53}$$

Therefore I conclude theta sweeps can provide enough compression to lift the timescale of the SR being learn by STDP from short synaptic timescales to relevant behavioural timescales on the order of seconds. Note this ballpark estimate is not intended to be precise, and doesn't account for many unknowns for example the covariability of sweep speed with running speed, variability of

sweep speed with track length or cell size which could potentially extend this range further.

C.9.3 Differences between STDP and TD learning: where my model doesn't work

I only drew a hand-waving connection between the TD-derived Hebbian learning rule in Equation (C.45) and STDP. There are numerous difference between STDP and TD learning, these include the fact that

1. Depression in Equation (C.45) is dependent on the dendritic voltage which is not true for my STDP rule.

2. Depression in Equation (C.45) is not explicitly dependent on the time between post and presynaptic activity, unlike STDP.

3. Equation (C.45) is a continuous learning rule for continuous firing rates, STDP is a discrete learning rule applicable only to spike trains.

Analytic comparison is difficult due to this final difference which is why in this chapter I instead opted for empirical comparison. My goal was never to derive a spike-time dependent synaptic learning rule which replicates TD learning, other papers have done work in this direction (see Brea et al. (2016) and Bono et al. (2021)), rather I wanted to (i) see whether unmodified learning rules measured to be used by hippocampal neurons perform and (ii) study whether phase precession aids learning. Under regimes tested here, STDP seems to hold up well.

These differences aside, the learning rule does share other similarities to my model set-up. A special feature of this learning rule is that it postulates that somatic voltage driving postsynaptic activity during learning isn't affected by the neurons own dendritic voltage. Rather, dendritic voltages affect the *plasticity* by setting the potentiation threshold. These learning rules have been studies under the collective name of "voltage dependent" Hebbian learning rules[CITE]. This matches the learning setting I use here where, during learning,

CA1 neurons are driven by one and only one CA3 feature (the "target feature") whilst the weights being trained W_{ij} don't immediately effect somatic activity during learning. The lack of online updating matches the electrophysiological observation that plasticity between CA3 and CA1 is highest during the phase of theta when CA1 is driven by Entorhinal cortex and lowest at the phase when CA3 actually drives CA1 Hasselmo et al. (2002).

Finally, there is one clear failure for my STDP model – learning very long timescale successor features. Unlike TD learning which can 'bootstrap' long timescale associations through intermediate connections, this is not possible with my STDP rule in its current form. Brea et al. (2016) Brea et al. (2016) and Bono et al. (2021) Bono et al. (2021) show how Equation (C.45) can be modified to allow long timescale SRs whilst still enforcing the timescale constraint I imposed in Equation (C.40) thus still maintaining the biological plausibility of the learning rule, this requires allowing the dendritic voltage to modify the somatic voltage during learning in a manner highly similar to bootstrapping in RL. Specifically in the former study this is done by a direct extension to the two-compartment model, in the latter it is recast in a one-compartment model although the underlying mathematics shares many similarities. Ultimately both mechanisms could be at play; even in neurons endowed with the ability to bootstrap long timescale association with short timescale plasticity kernels phase precession would still increase learning speed significantly by reducing the *amount* of bootstrapping required by a factor of κ_θ, something I intend to study more in future work. Finally it isn't clear what timescales predictive encoding in the hippocampus reach, there is likely to be an upper limit on the utility of such predictive representations beyond which the animal use model-based methods to find optimal solution which guide behaviour.

C.10 Supplementary Analyses and Ablations

C.10.1 Cell Size and Agent Speed Effects

For convenience, panel a of Figure C.1 duplicates the experiment shown in Figure 3.2a-e. The only change is learning time was extended from 30 minutes to 1 hour.

Movement speed variability Panel b shows an experiment where I reran the simulation shown in Figure 3.2a-e except, instead of a constant motion speed, the agent moves with a variable speed drawn from a continuous stochastic process (an Ornstein Uhlenbeck process). The parameters of the process were selected so the mean velocity remained the same (16 cm s^{-1} left-to-right) but now with significant variability (standard deviation of 16 cm s^{-1} thresholded so the speed can't go negative). Essentially, the velocity takes a constrained random walk. This detail is important: the velocity is not drawn randomly on each time step since these changes would rapidly average out with small dt, rather the change in the velocity (the acceleration) is random - this drives slow stochasticity in the velocity where there are extended periods of fast motion and extended periods of slow motion. After learning there is no substantial difference in the learned weight matrices. This is because both TD and STDP learning rules are able to average-over the stochasticity in the velocity and converge on representations representative of the mean statistics of the motion.

Smaller place cells and faster movement Nothing fundamental prevents learning from working in the case of smaller place fields or faster movement speeds. I explore this in Figure C.1, panel c, as follows: the agent speed is doubled from 16 cm s^{-1} to 32 cm s^{-1} and the place field size is shrunk by a factor of 5 from 2 m diameter to 40 cm diameter. To facilitate learning I also increase the cell density along the track from 10 cells m^{-1} to 50 cells m^{-1}. I also shrink the track size from 5 m to 2 m (any additional track is redundant due to the circular symmetry of the set-up and small size of the place cells). I then train for 12 minutes. This time was chosen since 12 minutes moving at 32 cm s^{-1} on a 2 m track means the same number of laps as 60 mins moving at

16 cm s^{-1} on a 5 m track (96 laps in total). Despite these changes the weight matrix converged with high similarity to the successor matrix with a shorter time horizon (0.5 s). Convergence time measured in minutes was faster than in the original case but this is mostly due to the shortened track length and increased speed. Measured in laps it now takes longer to converge due to the decreased number of spikes (smaller place fields and faster movement through the place fields). This can be seen in the shallower convergence curve, panel c (right) relative to panel a.

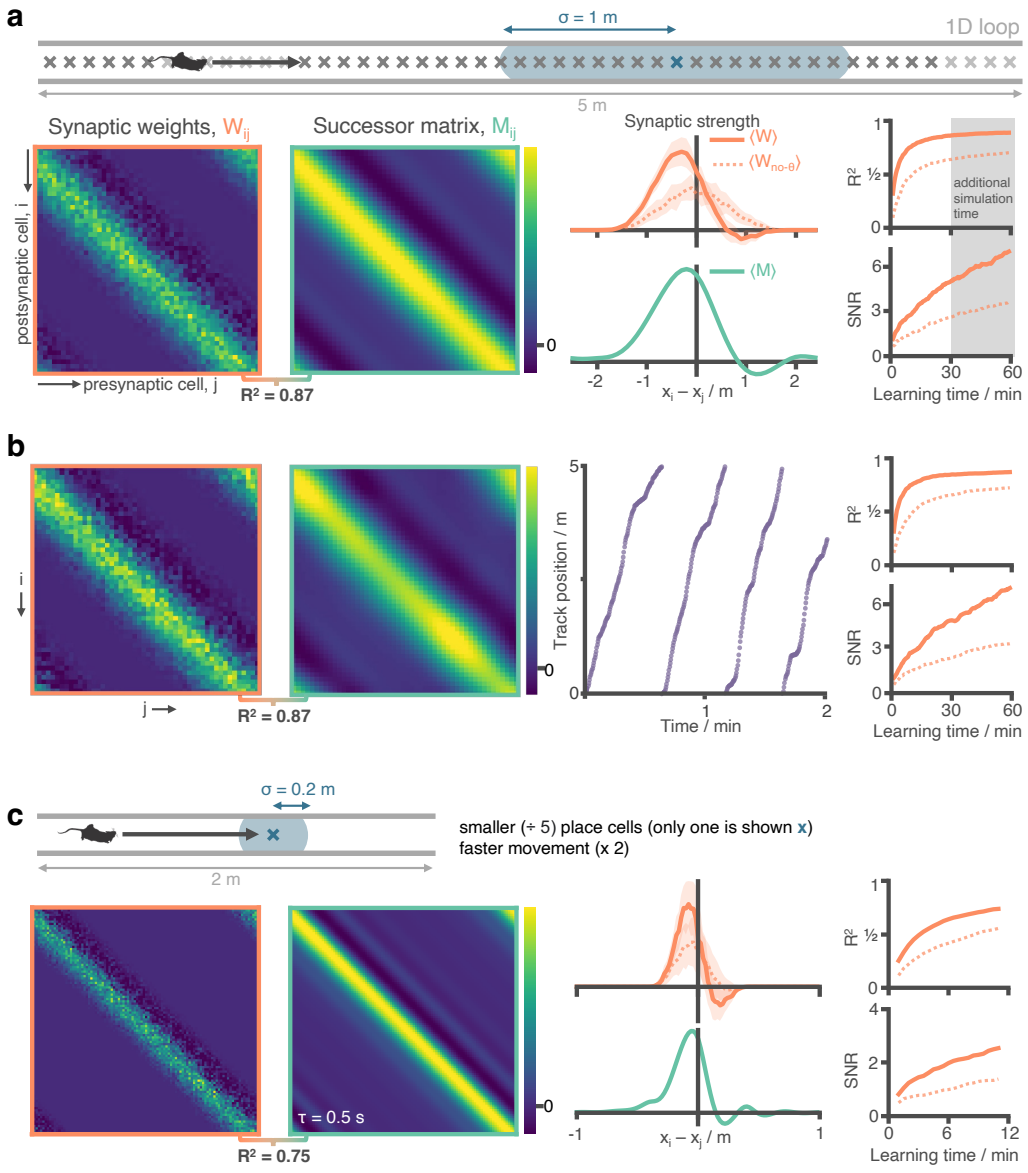

Figure C.1: STDP and phase precession combine to make a good approximation of the SR independent of place cell size and running speed statistics. **a** Figure 3.2 panels a-e have been repeated (additional 30 minutes simulation carried out) for ease of comparison. **b** I repeat the experiment with non-uniform running speed. Here, running seed is sampled according to a continuous stochastic process (Ornstein Uhlenbeck) with mean of 16 cm s^{-1} and standard deviation 16 cm s^{-1} thresholded to prevent negative speeds. As can be seen in the trajectory figure speed varies smoothly but significantly, including regions where the agent is almost stationary. Despite this there is no observable difference to the synaptic weights after learning. **c** I reduce the place cell diameter from 2 m to 0.4 m (5x decrease) and increase the motion speed from 16 cm s^{-1} to 32 cm s^{-1} (2x increase). I increase the cell density along the track from 10 cells m^{-1} to 50 cells m^{-1} to preserve cell overlap density. To reduce the computational load of training I shrink the track length from 5 m to 2 m (any additional track is symmetric and redundant when place cells are this small anyway). Note the adjusted training time: 12 minutes on a 2 m track at 32 cm s^{-1} corresponds to the same number of laps as 60 min on a 5 m track at 16 cm s^{-1} as shown for comparison in panel (a). Under these conditions the STDP + phase precession learning rule well approximates the successor features with a shorter time horizon of $\tau = 0.5$.

C.10.2 Weight Initialisation and Update Schedule

Random initialisation: In Figure C.2, panel a, I explore what happens if weights are initialised randomly. Rather than the identity, the weight matrix during learning is fixed ("anchored") to a sparse random matrix $\mathsf{W}^{\mathsf{A}}_{ij}$; this is defined such that each CA1 neuron receives positive connections from 3, 4 or 5 randomly chosen CA3 neurons with weights summing to one. In all other respects learning remains unchanged. CA1 neurons now have multi-modal receptive fields since they receive connections from multiple, potentially far apart, CA3 cells. This shouldn't cause a problem since each sub-field now acts as its own place field phase precessing according to whichever place cells in CA3 is driving it. Indeed it doesn't: after learning with this fixed but random CA3-CA1 drive, the synaptic weights are updated on aggregate and compares favourably to the successor matrix (panel a, middle and right). Specifically this is the successor matrix which maps the unmixed uni-modal place cells in CA3 to the successor features of the new multi-modal "mixed" features found in CA1 before learning. I note in passing that this is easy to calculate due to the linearity of the successor feature (SF): a SF of a linear sum of features is equal to a linear sum of SF, therefore I can calculate the new successor matrix using

the same algorithm as discussed above then rotating it by the sparse random matrix, $\mathsf{M}'_{ij} = \sum_k \mathsf{W}^{\mathsf{A}}_{ik}\mathsf{M}_{kj}$.

In order that some structure is visible matrix rows (which index the CA1 postsynaptic cells) have been ordered according to the location of the CA1 peak activity. This explains why the random sparse matrix (panel a, middle) looks ordered even though it isn't. After learning the STDP successor feature looks close in form to the TD successor feature and both show a shift and skew backwards along the track (panel a, rights, one example CA1 field shown).

Online weight updating: In Figure C.2, panels b, c and d, I explore what happens if the weights are updated online during learning. It is not possible to build a stable fully online model (as I suspect the review realised) and it is easy to understand why: if the weight matrix doing the learning is also the matrix doing the driving of the downstream features then there is nothing to prevent instabilities where, for example, the downstream feature keeps shifting backwards (no convergence) or the weight matrix for some/all features disappears or blows up (incorrect convergence). However it is possible to get most of the way there by splitting the driving weights into two components. The first and most significant component is the STDP weight matrix being learned online, this creates a "closed loop" where changes to the weights affects the downstream features which in turn affect learning on the weights. The second smaller component is what I call the "anchoring" weights, which I set to a fraction of the identity matrix (here $\frac{1}{2}$) and are not learned. In summary, Equation (C.13) becomes

$$\tilde{\psi}_i(\mathbf{x}, t) = \sum_j \left(\mathsf{W}_{ij}(t) + \mathsf{W}^{\mathsf{A}}_{ij}\right) f_j(\mathbf{x}, t) \tag{C.54}$$

for $\mathsf{W}^{\mathsf{A}}_{ij} = \frac{1}{2}\delta_{ij}$.

These anchoring weights provide structure, analogous to a target signal or "scaffold" onto which the successor features will learn without risk of infinite backwards expansion or weight decay. After learning when analysing the

weight/successor features the anchoring component is not considered.

Every other model of TD learning implicitly or explicitly has a form of anchoring. For example in classical TD learning each successor feature receives a fixed "reward" signal from the feature it is learning to predict (this is the second term in Equation (C.23)). Even other "synaptically plausible" models include a non-learnable constant drive (see Bono et al. (2021) CA3-CA1 model, more specifically the bias term in their Eqn. (12)). This is the approach taken here. I add the additional constraint that the sum of each row of the weight matrix must be smaller than or equal to 1, enforced by renormalisation on each time step. This constraint encodes the notion that there may be an energetic cost to large synaptic weight matrices and prevents infinite growth of the weight matrix.

$$\mathsf{W}_{ij}(t) \leftarrow \frac{\mathsf{W}_{ij}(t)}{\max(1, \sum_j \mathsf{W}_{ij})} \tag{C.55}$$

The resulting evolution of the learnable weight component, $\mathsf{W}_{ij}(t)$, is shown in panel b (middle shows row aligned averages of $\mathsf{W}_{ij}(t)$ from t=0 minutes to t = 64 minutes, on the full matrices are shown) and panel f (full matrix) from being initialised to the identity. The weight matrix evolves to look like a successor matrix (long skew left of diagonal, negative right of diagonal). One risk, when weights are updated online, is that the asymmetric expansion continues indefinitely. This doesn't happen and the matrix stabilises after 15 minutes (panel e, color progression). It is important to note that the anchoring component is smaller than the online weight component and I believe it could be made very small in the limit of less noisy learning (e.g. more cells or higher firing rates).

In panel c I explore the combination: random weight initialisation *and* online weight updating. As can be seen, even with rather strong random initial weights learning eventually "forgets" these and settles to the same successor matrix form as when identity initialisation was used.

In panel d I show that anchoring *is* essential. Without it ($\mathsf{W^A}_{ij} = 0$) the weight matrix initially shows some structure shifting and skewing to the left

but this quickly disintegrates and no observable structure remains at the end of learning.

Many-to-few spiking model In Figure C.2, panel e, I simulate the more biologically realistic scenario where each CA1 neuron integrates spikes (rather than rates) from a large (rather than equal) number of upstream CA3 neurons. This is done with two changes:

Firstly I increased the number of CA3 neurons from 50 to 500 while keeping the number of CA1 neurons fixed. Each CA1 neuron now receives fixed anchoring drive from a Gaussian-weighted sum of the 10 (as opposed to 1) closest CA3 neurons.

Secondly, since in my standard model spikes are used for learning but neurons communicate via their rates, I change this so that CA3 spikes directly drive CA1 spikes in the form of a reduced spiking model. Let $\mathsf{X}^{\mathsf{CA1}}_{i,\mathsf{t}}$ be the spike count of the i^{th} CA1 neuron at timestep t and $\mathsf{X}^{\mathsf{CA3}}_{j,\mathsf{t}}$ the equivalent for the j^{th} CA3 neuron then, under the reduced spiking model,

$$\mathsf{Pr}(\mathsf{X}^{\mathsf{CA1}}_{i,\mathsf{t}} = k) = \mathsf{Poisson}(k, \lambda_{i,\mathsf{t}}) \tag{C.56}$$

$$\lambda_{i,\mathsf{t}} = \frac{1}{dt} \sum_j \mathsf{W}^{\mathsf{A}}{}_{ij} \mathsf{X}^{\mathsf{CA3}}_{j,\mathsf{t}} \tag{C.57}$$

As can be expected, this model is very similar to the original model since CA3 spikes are noisy sample of their rates. This noise should average out over time and the simulations indeed confirm this.

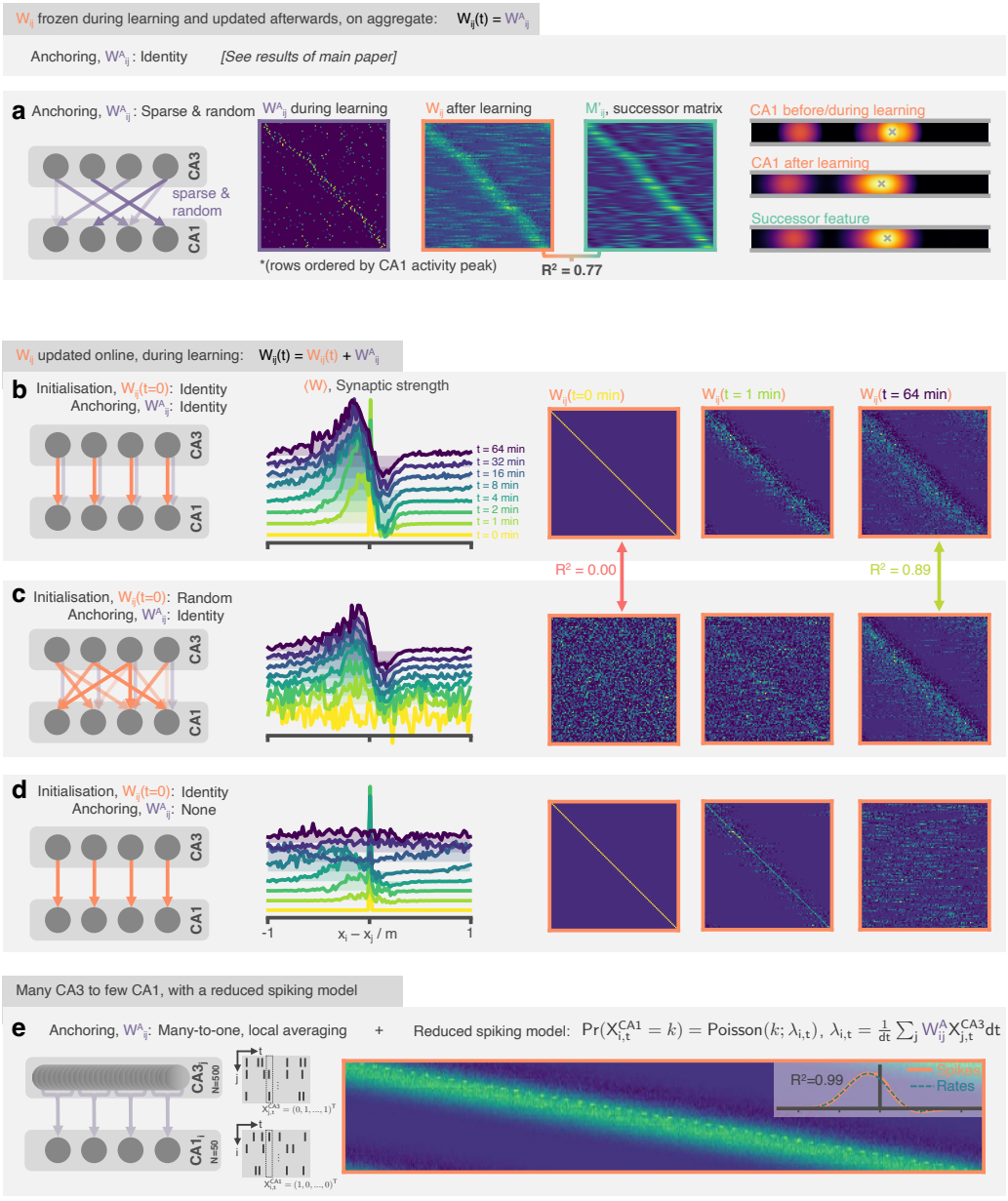
W_{ij} frozen during learning and updated afterwards, on aggregate: $W_{ij}(t) = W^A_{ij}$
Anchoring, W^A_{ij}: Identity [See results of main paper]
a Anchoring, W^A_{ij}: Sparse & random
W^A_{ij} during learning
W_{ij} after learning
M'$_{ij}$, successor matrix
CA1 before/during learning
CA1 after learning
Successor feature
CA3
sparse & random
CA1
*(rows ordered by CA1 activity peak)
R^2 = 0.77
W_{ij} updated online, during learning: $W_{ij}(t) = W_{ij}(t) + W^A_{ij}$
b Initialisation, $W_{ij}(t=0)$: Identity
Anchoring, W^A_{ij}: Identity
⟨W⟩, Synaptic strength
t = 64 min
t = 32 min
t = 16 min
t = 8 min
t = 4 min
t = 2 min
t = 1 min
t = 0 min
W_{ij}(t=0 min)
W_{ij}(t = 1 min)
W_{ij}(t = 64 min)
R^2 = 0.00
R^2 = 0.89
c Initialisation, $W_{ij}(t=0)$: Random
Anchoring, W^A_{ij}: Identity
d Initialisation, $W_{ij}(t=0)$: Identity
Anchoring, W^A_{ij}: None
-1
$x_i - x_j$ / m
1
Many CA3 to few CA1, with a reduced spiking model
e Anchoring, W^A_{ij}: Many-to-one, local averaging
+
Reduced spiking model: $\Pr(X^{CA1}_{i,t} = k) = \text{Poisson}(k; \lambda_{i,t})$, $\lambda_{i,t} = \frac{1}{dt}\sum_j W^A_{ij} X^{CA3}_{j,t} dt$
CA3$_j$ N=500
CA1$_i$ N=50
R^2=0.99
Spikes
Rates

Figure C.2: The STDP and phase precession model learns predictive maps irrespective of the weight initialisation and the weight updating schedule. In the original model weights are set to the identity before learning and kept ("anchored") there, only updated on aggregate after learning. In these panels I explore variations to this set-up. **a** (Left) Weights are anchored to a sparse random matrix, not the identity. (Middle) Three weight matrices show the random weights before/during learning, the weights once they have been updated on aggregate after learning and the successor matrix corresponding to the successor features of the mixed features. Matrix rows are ordered by peak CA1 activity location in order that some structure is visible. (Right) An example CA1 feature (top) before learning and (middle) after learning alongside (bottom) the corresponding successor feature. **b** (Left) The weight matrix is no longer fixed during learning, instead it is initialised to the identity and updated online during learning. A fixed component (0.5 x δ_{ij}) is added to "anchor" the downstream representations. (Middle and right) After learning the STDP weights show an asymmetric shift and skew against the direction of motion and a negative band ahead of the diagonal just as was observed for successor matrices and the fixed weight model. This backwards expansion does not carry on extending indefinitely (a risk when the weights are updated online) but stabilises. **c** Like panel b but weights are randomly initialised. After learning the weights have "forgotten" their initial structure and are essentially identical to in the case of identity initialisation. **d** Like panel b except no anchoring weights are added. Now there is no fixed component anchoring CA1 representations, structure in the synaptic weights rapidly disintegrates. **e** 500 CA3 neurons drive 50 CA neurons where each CA1 neuron is anchored to a Gaussian-weighted sum of 10 closest CA3 cells. CA3 spikes now directly drive CA1 spikes according to a reduced spiking model. The inset shows the row-averages and a comparison to the result for an equivalent simulation with the rate-model used in the rest of this chapter.

C.10.3 Hyperparameter Sweep

I perform a hyperparameter sweep over STDP and phase precession parameters to see which are optimal for learning successor matrices. Remarkably the optimal parameters (those giving highest R^2 between the weight matrix and the successor matrix) are found to be those – or vary close to those – used by biological neurons (Figure C.3). Specifically, to avoid excess computational costs two independent sweeps were run: the first was run over the four relevant STDP parameters (the two synaptic plasticity timescales, the ratio of potentiation to depression and the firing rate) and the second was run over the phase precession parameters (phase precession spread parameter and the phase precession fraction).

On all cases the optimal parameter sits close to the biological parameter I used in this chapter (panel c, d). One exception is the firing rate where higher

firing rates always give better scores, likely due to the decreased effect of noise, however it is reasonable biology can't achieve arbitrarily high firing rates for energetic reasons.

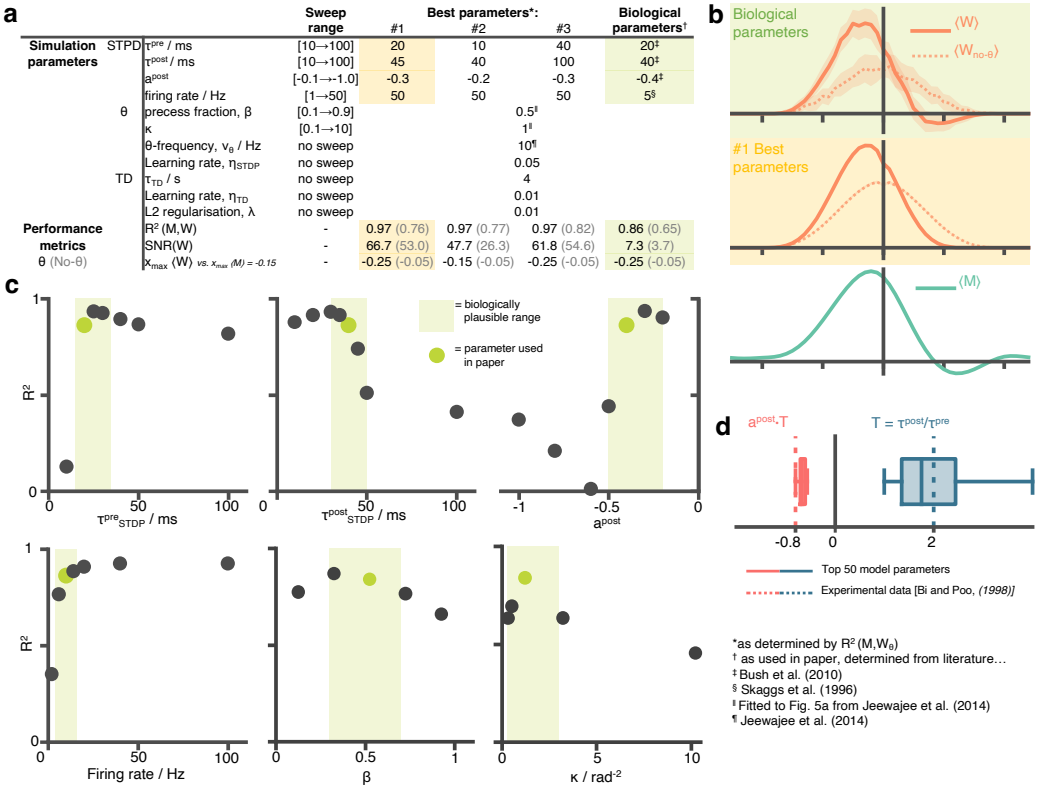

Figure C.3: A hyperparameter sweep over STDP and phase precession parameters shows that biological parameters are suffice, and are near-optimal for approximating the successor features **a** A table showing all parameters used in this chapter and the ranges over which the hyperparameter sweep was performed. For each parameter setting I estimate performance metrics to judge whether the STDP parameters do well at learning the successor features. **b** Visually inspecting the row aligned STDP weight matrices I see the optimal parameters do not significantly outperform the biologically chosen ones. Although the optimal parameter setting results in a slightly higher R^2, they fail to capture the right-of-centre negative weights present in the TD successor matrix, unlike the biological ones. **c** Slices through the parameter sweep hypercube. For each plot, parameter values of the other five variables are fixed to the green values (i.e. are the ones used in this chapter). **d** The top 50 performing parameter combination are stored and box plots for the conjugate parameter $T = \frac{\tau^{\text{pre}}}{\tau^{\text{post}}}$, the ratio of time windows for potentiation and depression, and $-a^{\text{post}} \cdot T$, effectively the ratio of the areas under the curve left and right of the y-axis on the STDP plot Figure 3.1c. In both cases the 'best parameters' include the true parameter values, measured experimentally by Bi and Poo (1998) Bi et al. (1998)

C.10.4 Phase Precession Hyperparameter Sweep

The optimality of biological phase precession parameters In Figure C.3 I ran a hyperparameter sweep over the two parameters associated with phase precession: κ, the von Mises parameter describing how noisy phase precession is and β, the fraction of the full 2π theta cycle phase precession crosses. The results show that for both of these parameters there is a clear "goldilocks" zone around the biologically fitted parameters I chose originally. When there is too much (large κ, large β) or too little (small κ, small β) phase precession performance is worse than at intermediate biological amounts of phase precession. Whilst – according to the central hypothesis of the chapter – it makes sense that weak or non-existence phase precession hinders learning, it is initially counter intuitive that strong phase precession also hinders learning.

I speculate the reason is as follows, when β is too big phase precession spans the full range from 0 to 2π, this means it is possible for a cell firing very late in its receptive field to fire just before a cell a long distance behind it on the track firing very early in the cycle because 2π comes just before 0 on the unit circle. When κ is too big, phase precession is too clean and cells firing at opposite ends of the theta cycle will never be able to bind since their spikes will never fall within a 20 ms window of each other. I illustrate these ideas in Figure C.4 by first describing the phase precession model (panel a) then simulating spikes from 4 overlapping place cells (panel b) when phase precession is weak (panel c), intermediate/biological (panel d) and strong (panel e). I confirm these intuitions about why there exists a phase precession "goldilocks" zone by showing the weight matrix compared to the successor matrix (right hand side of panels c, d and e). Only in the intermediate case is there good similarity.

Phase precession of CA1 In most results shown in this chapter the weights are anchored to the identity during learning. This means each CA1 cells inherits phase precession from the one and only one CA3 cell it is driven by. It is important to establish whether CA1 still shows phase precession *after* learning

when driven by multiple CA3 cells or, equivalently, during learning when the weights aren't anchored and it is therefore driven by multiple CA3 neurons. Analysing the spiking data from CA1 cells after learning (phase precession turned on) shows it does phase precession. This phase precession is noisier than the phase precession of a cell in CA3 but only slightly and compares favourably to real phase precession data for CA1 neurons (panel f, right, adapted from Jeewajee et al. (2014) Jeewajee et al. (2014)).

The reason for this is that CA1 cells are still localised and therefore driven mostly by cells in CA3 which are close and which peak in activity together at a similar phase each theta cycle. As the agent moves through the CA1 cell it also moves through all the CA3 cells and their peak firing phase precesses driving an earlier peak in the CA1 firing. Phase precession is CA1 after learning is noisier/broader than CA3 but far from non-existent and looks similar to real phase precession data from cells in CA1.

Phase shift between CA3 and CA1 In Figure C.4g I simulate the effect of a decreasing phase shift between CA3 and CA1. As observed by Mizuseki et al. (2012) Mizuseki et al. (2012) there is a phase shift between CA3 and CA1 neurons starting around 90 degrees at the end of each theta cycle (where cells fire as their receptive field is first entered) and decreasing to 0 at the start. I simulate this by adding a temporal delay to all downstream CA1 spikes equivalent to the phase shifts of 0°, 45° and 90°. The average of the weight matrices learned over all three examples still displays clear SR-like structure.

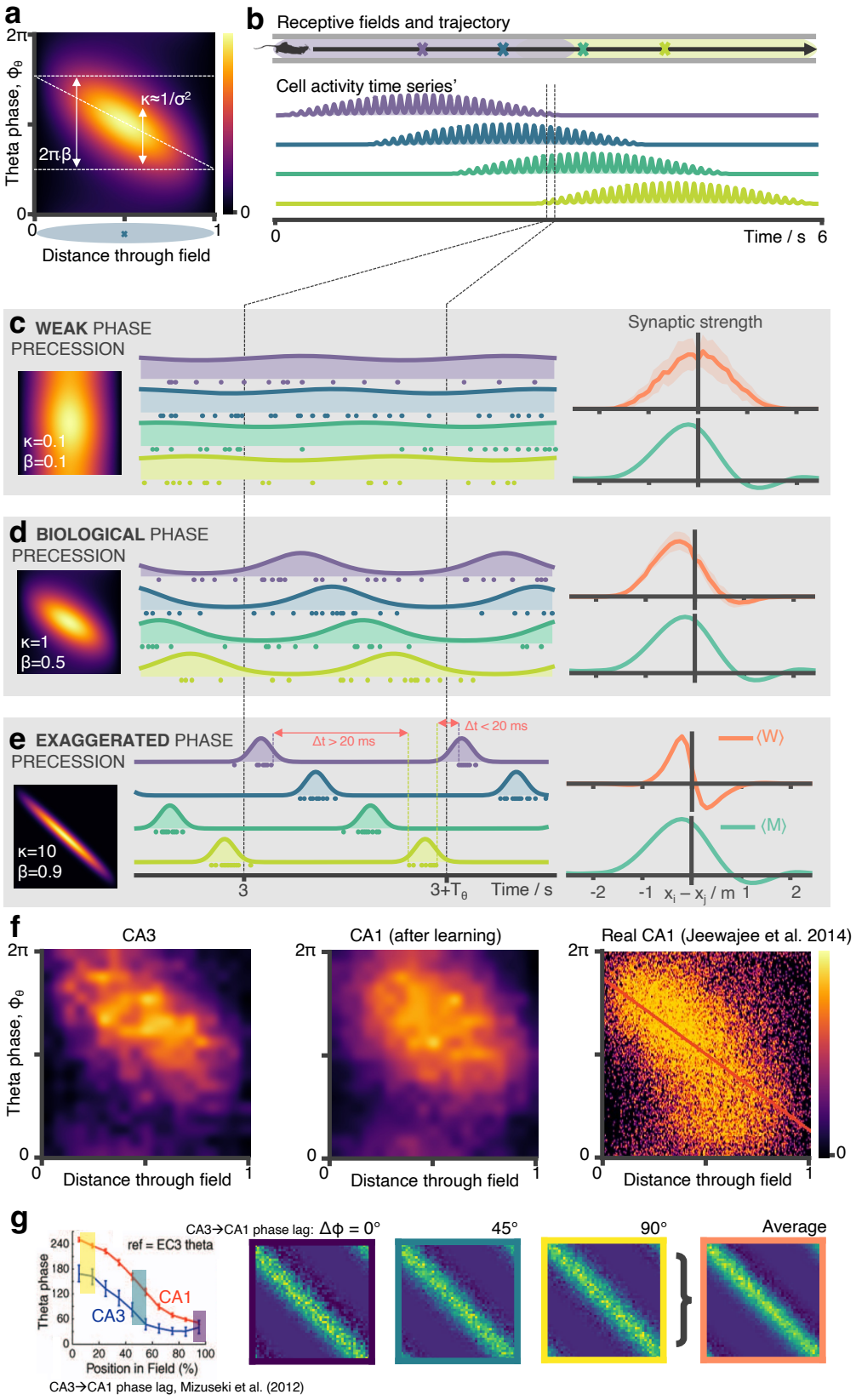
a
2π
Theta phase, φθ
κ≈1/σ²
2πβ
0
0
1
Distance through field
b
Receptive fields and trajectory
Cell activity time series'
0
Time / s 6
c WEAK PHASE PRECESSION
κ=0.1
β=0.1
Synaptic strength
d BIOLOGICAL PHASE PRECESSION
κ=1
β=0.5
e EXAGGERATED PHASE PRECESSION
Δt > 20 ms
Δt < 20 ms
κ=10
β=0.9
3
3+Tθ
Time / s
⟨W⟩
⟨M⟩
-2
-1
xi – xj / m
1
2
f
CA3
CA1 (after learning)
Real CA1 (Jeewajee et al. 2014)
2π
Theta phase, φθ
0
Distance through field
1
g
CA3→CA1 phase lag: Δφ = 0°
45°
90°
Average
ref = EC3 theta
Theta phase
CA1
CA3
Position in Field (%)
CA3→CA1 phase lag, Mizuseki et al. (2012)

Figure C.4: Biological phase precession parameters are optimal for learning the SR. **a** I model phase precession as a von Mises centred at a preferred theta phase which precesses in time. This factor modulates the spatial firing field. It is parameterised by κ (von Mises width parameter, aka noise) and β (fraction of full 2π phase being swept, diagonal line). I showed in a previous figure that biological phase precession parameters are optimal. Any more or less phase precession degrades performance. It is easy to understand why: **b** Consider four place cells on a track (purple, blue, green, yellow) where the first and last just overlap. **c** In the weak phase precession regime there is no ordering to the spikes and STDP can't learn the asymmetry in the successor matrix (right) **d** In the medium phase precession regime spikes are broadly ordered in time (purple then blue then green...) so the symmetry is broken and STDP learns a close approximation the successor matrix **e**) In the "exaggerated" phase precession regime there exist two problems for learning SRs: "causal" bindings (e.g. from presynaptic purple to postsynaptic yellow, which sits in front of purple) are inhibited for anything except the most closely situated cell pairs due to the sharp tuning curves. Secondly, though this is a less important effect, when β is too large it is possible for incorrect "acausal" bindings to be formed due to one cell (e.g. yellow) firing late in theta cycle N just before another cell located far behind it on the track fires (e.g. purple) in theta cycle N+1. **f** CA1 cells will phase precess when driven by multiple CA3 place cells. Here I show phase precession (spike probability for different theta phases against distance travelled through field) for CA3 basis features and CA1 STDP successor features after learning. Although noisier there is still a clear tendency for CA1 cells to phase precess. Real CA1 cell phase precession can be 'noisy'; I show for comparison a phase precession plot for CA1 place field taken from Jeewajee et al. (2014), the same data to which the parameters were fitted. The schematic simulation figures showing spiking phase precession data in panels b, c, d and e were made using an open source hippocampal data generation toolkit George et al. (2024). Figure 2 –figure supplement 4 panel f, right has been adapted from Figure 5a from Jeewajee et al. 2014. **g** (Left) A decreasing phase shift is measured between CA3 and CA1, starting from 90° late in the cycle – the phase cells initially spike at as animals enter a field – and ending at 0° early in the cycle, panel is adapted from Mizuseki et al. 2012. (Middle) Three phase shifts (0°, 45° and 90°) are simulated and the average of the resulting synaptic weight matrices is taken (right)

Appendix D

Appendix to Chapter 4

D.1 Code Availability

Code to generate the results in Chapter 4 is available at `https://github.com/TomGeorge1234/HelmholtzHippocampus`.

D.2 Detailed Model Implementation

A general description of the model is given here. Specifics for each experiment (i.e. learning rates, layer sizes, time constants etc.) are given in later sections.

D.2.1 Dendritic Updates

Complete versions of the dendritic update rules (summarised in eqs. (4.2) and (4.3)) are given below. It is assumed that dendrites receive and integrate synaptic inputs according to the following dynamics:

$$\left.\begin{aligned} \tau \frac{d\mathbf{p}_B(t)}{dt} &= -\mathbf{p}_B(t) + \bar{\mathbf{p}}(z(t)) \\ \tau \frac{d\mathbf{g}_B(t)}{dt} &= -\mathbf{g}_B(t) + \sigma_{g_B}(\mathbf{w}_{g_B}\mathbf{p}(t)) \end{aligned}\right\} \quad \text{Inference model} \tag{D.1}$$

$$\left.\begin{aligned} \tau \frac{d\mathbf{g}_A(t)}{dt} &= -\mathbf{g}_A(t) + \sigma_{g_A}(\mathbf{w}_{g_A}\mathbf{g}(t)) \\ \tau \frac{d\mathbf{p}_A(t)}{dt} &= -\mathbf{p}_A(t) + \sigma_{p_A}(\mathbf{w}_{p_A}\mathbf{g}(t)) \end{aligned}\right\} \quad \text{Generative model.} \tag{D.2}$$

These dynamics are discretised in order to be implemented computationally by making the common assumption that neural dynamics are fast ($\tau \approx 0$ ms) relative to the timescale of the synaptic inputs and so the compartments are

always at equilibrium, recovering eqs. (4.2) and (4.3). This is valid in the regime where the environmental latent updates slowly compared to neural timescales. The notation used here admits the possible presence of biases as well as the weights (though biases typically aren't used) by assuming a row of constant 1's could be added to the synaptic inputs effectively absorbing a bias into the weight matrix without loss of generality, for example $\mathbf{w}_{g_B}\mathbf{p}(t) \leftarrow \mathbf{w}_{g_B}\mathbf{p}(t) + b_{g_B}$.

D.2.2 Somatic Updates

Somatic updates rules (eqs. (4.4) and (4.5)) are repeated here for completeness:

$$\begin{aligned}\mathbf{p}(t) &= \theta(t)\mathbf{p}_B(t) + (1-\theta(t))\mathbf{p}_A(t)\\ \mathbf{g}(t) &= \theta(t)\mathbf{g}_B(t) + (1-\theta(t))\mathbf{g}_A(t).\end{aligned} \tag{D.3}$$

where $\theta(t)$ is a 5 Hz global theta oscillation variable defined by the square wave function:

$$\theta(t) = \begin{cases} 1, & \text{if } t/T \mod 1 \leq 0.5 \\ 0, & \text{if } t/T \mod 1 > 0.5 \end{cases} \tag{D.4}$$

D.2.3 Update Ordering

For this hierarchical network of multicompartmental neurons, the order in which these discrete updates are performed to the different layers and the different compartments within these layers must be specified. Strictly speaking, when the discretisation timestep dt is small this ordering is arbitrary, but I include it here for completeness.

I update the layers from bottom to top: first, I update the latent or "environment" and increment the global clock ($z(t+dt) \leftarrow z(t)$ & $t + dt \leftarrow t$). Next I update both dendritic compartments of the sensory layer ($\mathbf{p}_B(t+dt) \leftarrow \mathbf{p}_B(t)$ & $\mathbf{p}_A(t+dt) \leftarrow \mathbf{p}_A(t)$ noting that it makes no difference in which order these updates are done as they are independent. Then I update the somatic compartment of the sensory layer ($\mathbf{p}(t+dt) \leftarrow \mathbf{p}(t)$). Next I work upwards to the hidden layer ($\mathbf{g}_B(t+dt) \leftarrow \mathbf{g}_B(t)$ & $\mathbf{g}_A(t+dt) \leftarrow \mathbf{g}_A(t)$ followed by $\mathbf{g}(t+dt) \leftarrow \mathbf{g}(t)$) then, if present, the topmost "conjunctive cells" are updated.

This gives the following dendritic update rules which are only slightly – and in the limit $dt \to 0$, irrelevantly – different from the simplified update rules given in the main text:

$$\begin{aligned} \mathbf{p}_B(t+dt) &= \bar{\mathbf{p}}(z(t+dt)) \\ \mathbf{p}_A(t+dt) &= \sigma_{p_A}(\mathbf{w}_{p_A}\mathbf{g}(t)) \\ \mathbf{g}_B(t+dt) &= \sigma_{g_B}(\mathbf{w}_{g_B}\mathbf{p}(t+dt)) \\ \mathbf{g}_A(t+dt) &= \sigma_{g_A}(\mathbf{w}_{g_A}\mathbf{g}(t)) \end{aligned} \tag{D.5}$$

D.2.4 Learning Rules

Learning rules are conceptually summarised by the equations given in the main text, eq. (4.6). The *full* equations are given here, which include some adjustments to account for the presence of non-linear activation functions and temporal smoothing of the local prediction error learning signals. In the multilayer network, all sets of learnable weights follow an equivalent learning rule. For this reason, it is given here in its most general form: Consider the synaptic weight w_{ij} connecting from the soma of presynaptic neuron j with activation f_j^{pre} to one of the dendritic compartments of a postsynaptic neuron i with activation $f_{C,i}^{\text{post}} = \sigma(V_{C,i}^{\text{post}})$ (this could be the basal or apical compartment, $C \in \{A, B\}$). Weights are updated on each timestep by an amount:

$$\delta w_{ij}(t) = \eta \text{PI}_{ij}(t) \tag{D.6}$$

where PI_{ij} is (following terminology used in Urbanczik et al. (2014)) the "plasticity induction" variable which is a low-pass filtered measure of the coincidence between the local prediction error and the synaptic input. The prediction error measures how far the activation of the dendritic compartment, $f_{C,i}^{\text{post}}$, is from the somatic activation f_i^{post}. In total, PI_{ij} is defined by the

following dynamics:

$$\tau_{\mathrm{PI}}\frac{d\mathrm{PI}_{ij}}{dt} = -\mathrm{PI}_{ij} + \underbrace{[f_i^{\mathrm{post}}(t) - f_{C,i}^{\mathrm{post}}(t)]}_{\text{postsynaptic prediction error}} \cdot \sigma'(V_{C,i}^{\mathrm{post}}(t)) \cdot \underbrace{f_j^{\mathrm{pre}}(t)}_{\text{presynaptic input}} \tag{D.7}$$

If the prediction error and one of the presynaptic inputs are both consistently large (i.e. over a time period $\mathcal{O}(\tau_{\mathrm{PI}})$) then the plasticity induction variable will therefore also be large and the weight connecting the pre- and postsynaptic neurons will be strengthened (thus decreasing future prediction errors). τ_{PI} is taken to be the same as used in Urbanczik et al. (2014), 100 ms. Note for fast filtering ($\tau_{\mathrm{PI}} \to 0$ ms) and linear activation functions this reduces to the simplified formulae given in the main text, eq. (4.6).

D.2.5 Synaptic Noise

I add synaptic noise to the dendritic activations. Each dendritic compartment maintains its own independent noise variable, $n(t)$, which is modelled as an Ornstein-Uhlenbeck process. The benefit of modelling neural noise with an Ornstein-Uhlenbeck process is that it is timestep size independent. The dynamics of the noise variable are given by:

$$n(t+dt) = n(t) + \frac{dt}{\tau}n(t) + \sqrt{\frac{2\sigma^2 dt}{\tau}}\xi(t) \tag{D.8}$$

where $\xi(t) \sim \mathcal{N}(0,1)$ is a white noise process. These dynamics lead to a stationary distribution of $n(t)$ which is Gaussian with zero mean and variance σ^2. The decorrelation timescale of the noise is τ. I fix $\tau = 300$ ms and $\sigma = 0.01$ Hz in order that noise is relatively slow and weak. Noise is added at each timestep to the activation of the dendrites, e.g. $\mathbf{p}_B(t) \to \mathbf{p}_B(t) + \mathbf{n}_B(t)$ where $\mathbf{n}_B(t)$.

D.2.6 Measuring Prediction Error

Figure 4.2b and fig. 4.3d show the prediction errors of the network layers decreasing throughout training. Here, the method for calculating these errors

is defined. A consequence of the learning rule is that during wake, the apical dendrites adjust to try minimise the discrepancy between the apical activation and the soma (which, during wake, is equal to the basal activation). During the sleep phase a short time later the basal dendrites adjust to try minimise the discrepancy between the basal activation and the soma (which, during sleep, is equal to apical activation). If learning is successful I would expect the apical and basal activations to converge, thus I use the following measures of the prediction error to track training performance in both layers of the network:

$$\begin{aligned}\mathcal{E}_p(t) &= \frac{1}{N_p}\sum_i |[\mathbf{p}_B(t)]_i - [\mathbf{p}_A(t)]_i| \\ \mathcal{E}_g(t) &= \frac{1}{N_g}\sum_i |[\mathbf{g}_B(t)]_i - [\mathbf{g}_A(t)]_i|.\end{aligned} \tag{D.9}$$

These are then smoothed with a decaying exponential kernel of timescale 60 seconds to remove some of the noise and better display the learning signal.

D.3 Relationship to Online Bayesian Inference

Bredenberg et al. (2021) derived local synaptic learning rules for a similar hierarchical network performing online latent inference starting from a loss function closely related to the evidence lower bound (ELBO) of variational inference. Here I will not repeat their derivation, instead I intend to highlight their starting point, the most important assumptions they made and the learning rules they derived, finally pointing out how theirs differ from the ones used here. The point is to demonstrate that the learning rules proposed are not arbitrary but can actually be derived from a more principled approach to online inference.

Bredenberg et al. (2021) consider a network receiving input from a latent variable z. The network has two layers, $\mathbf{p}_t$ and $\mathbf{g}_t$.[1] The network is trained to perform online inference over a sequence of observations from the environment,

[1]For convenience, their variables have been translated into the notation of this work ($\mathbf{g} \leftrightarrow \mathbf{r}$,$\mathbf{p} \leftrightarrow \mathbf{s}$,$\grave{} \leftrightarrow \breve{}$, $\mathbf{w} \leftrightarrow \grave{}$) so it is easier to compare.

$z_{0:T}$. To do this they start from the loss function

$$\mathcal{L} = \mathbb{E}_{\theta,\mathbf{z}}\Big[D_{KL}(\tilde{q}_w \parallel \tilde{p}_w)\Big] \tag{D.10}$$

where $\tilde{q}_w$ and $\tilde{p}_w$ are the following probability distributions over the layer variables $\mathbf{p}_t$ and $\mathbf{g}_t$:

$$\tilde{q}_w = \prod_{t=0}^{T} \left(q(\mathbf{g}_t|\mathbf{p}_t; w_{\text{inf}})p(\mathbf{p}_t|z_t)\right)^{\theta_t} p_m(\mathbf{g}_t, \mathbf{p}_t|\mathbf{p}_{t-1}, \theta_t; w_{\text{gen}})^{1-\theta_t}, \tag{D.11}$$

$$\tilde{p}_w = \prod_{t=0}^{T} \underbrace{\left(p(\mathbf{g}_t|\mathbf{p}_t; w_{\text{inf}})p(\mathbf{p}_t|z_t)\right)}_{\text{inference model}}^{1-\theta_t} \underbrace{p_m(\mathbf{g}_t, \mathbf{p}_t|\mathbf{p}_{t-1}, \theta_t; w_{\text{gen}})}_{\text{generative model}}^{\theta_t} \tag{D.12}$$

and $\theta_t \in \{0, 1\}$ is a binary variable (in their analysis they fix this to oscillate in fixed symmetric phases, e.g. 000111000111...). The two probability distributions, $\tilde{q}_w$ & $\tilde{p}_w$, which this loss function attempts to make similar to one another, can be interpreted as the probabilities over the layer variables $\mathbf{p}_t$ and $\mathbf{g}_t$ in two noisy neural networks[2] connected as I drew in Figure 4.1a: the first network alternates between phases of inference, where information flows bottom up from the latents z to the hidden layer $\mathbf{g}$, and generation, the opposite (inference-generation-inference-generation...), the second network alternates in exact counterphase (generation-inference-generation-inference...). This loss is a generalisation of the widely used evidence lower bound (ELBO) which corresponds to the case where $\theta_t = 1$ for all t. ELBO loss functions seek to make the inference and generative distributions over sensory and hidden variables similar. I will not delve further into the justifications for these types of loss functions other than to state that they are widely used (Kingma et al. 2022).

One of the key conceptual steps taken by Bredenberg et al. (2021) (and now myself) is to note that processes of performing inference and generation can

[2]Note there isn't *actually* two networks being trained. Instead they use a mathematical trick, deriving from the symmetry in the alternating phase of the theta cycle, to do away with the need to sample from both networks meaning they can derive local learning rules which can train a single network, e.g. $\tilde{q}_m$, on its own. This single network, like mine, contains both inference and generative models, represented by the two terms in eq. (D.11)

locally occur simultaneously as long as they are received into distinct dendritic compartments. The choice of which dendrite gates the soma (i.e. eq. (4.4)) then dictates the global state (wake or sleep) of the network. It also means, as they show, that the loss can be approximately optimized using local learning rules by comparing the dendritic compartment activation to that of the soma. The learning rules they derive, again translated into my notation, are as follows (note for simplicity I assume all activations are linear since non-linearities add only one additional multiplicative term into their update equations, see equations (14), (15) and (16) in (Bredenberg et al. 2021)):

$$\frac{d\mathbf{w}_{g_B}}{dt} \propto (1-\theta_t)(\mathbf{g}_t - \mathbf{g}_{B,t})\mathbf{p}_t^\mathsf{T}$$

$$\frac{d\mathbf{w}_{p_A}}{dt} \propto \theta_t(\mathbf{p}_t - \mathbf{p}_{A,t})\mathbf{g}_t^\mathsf{T} \tag{D.13}$$

$$\frac{d\mathbf{w}_{g_A}}{dt} \propto \theta_t(1-k_t)(\mathbf{g}_t - \mathbf{g}_{A,t-1})\mathbf{g}_{t-1}^\mathsf{T} \tag{D.14}$$

where $k_t = (1 - \delta(\theta_t - \theta_{t-1}))\theta_t$ is a term which is 1 if and only if $\theta_t = 1$ and $\theta_{t-1} = 0$ therefore it briefly turns off learning upon switching from sleep to wake.

Readers may like to compare these learning rules to the ones used here, as given in the main text eq. (4.6). The learning rules used here differ from theirs in the following way:

- I relax their discrete time assumption, opting for a continuous time formulation ($\mathbf{p}_t \to \mathbf{p}(t)$ etc.).
- I note that the terms in the equations proportional to θ_t or $1-\theta_t$ which actively turn on or off learning depending on whether $\theta_t = 0$ or 1 are unnecessary since the prediction error term naturally falls to zero anyway. For example, in eq. (D.13) when $\theta_t = 0$ the network is in sleep and so $\mathbf{p}_t = \mathbf{p}_{A,t}$. In this case the prediction error is zero by definition and learning ceases even without the preceding θ_t term.
- I disregard the $1-k_t$ term. Empirically this does not seem to damage the

model and theoretically its impact should only be small in my continuous time formulation where the network is only switching from sleep to wake for a negligible proportion of the time.

- Upon provisional theoretical and experimental justification, θ is likened to the theta component of the hippocampal local field potential and set to 5 Hz.

Ultimately these changes are surface level. The learning rules used here can – and should – be understood as a close approximation to those derived by Bredenberg et al. (2021). Consequently it is appropriate to consider my hippocampal model as learning to perform approximately optimal online Bayesian inference.

D.4 Artificial Task: Implementation Details

$N_z = 5$ independent, autocorrelated, random latent variables are sampled from a Gaussian process with a squared exponential covariance function of width 1 second, samples of these are shown in Figure 4.2a and Figure D.2. The sensory layer is large ($N_p = 50$) relative to the compressed hidden layer ($N_g = N_z = 5$) and receives a random mixture of the latents into the basal compartments as described in the text. All activation functions are linear, no layers have biases, all learning rates are set to $\eta = 0.01$, and the discretisation timestep was $dt = 25$ ms. Weights are initialised randomly $[\mathbf{w}_{\mathbf{g}_B}]_{ij} \sim \mathcal{N}(0, 1/\sqrt{N_p})$, $[\mathbf{w}_{\mathbf{p}_A}]_{ij} \sim \mathcal{N}(0, 1/\sqrt{N_g})$, $[\mathbf{w}_{\mathbf{g}_A}]_{ij} \sim \mathcal{N}(0, 0.1/\sqrt{N_g})$ where the smaller initialisation on the recurrent weights, $\mathbf{w}_{g_A}$, was chosen to prevent unstable dynamics.

Before learning – since weights are initialised randomly – basal and apical voltages in the sensory layer are unmatched when tested for a period in wake mode (Figure D.1a). When tested for a period in sleep mode, the small initialisation of the recurrent weights means the hidden layer cannot sustain activity (Figure D.1b, top) which decays and decorrelates rapidly in contrast to the true latents (Figure D.1c). Compare this to after learning where, during wake, basal and apical voltages in the sensory layer are closely matched implying

accurate autoencoding through the compressed hidden layer. During sleep, the hidden layer generates sustained activity statistically similar to the true latents (they do not match because during sleep the true latents are not driving the network; even during wake it would only be expected that the network represent the true latents in its latent space up to a linear rotation), i.e. it is functioning as a generative model. Note the only source of randomness driving stochasticity and activity in the network is the noise in the dendritic updates themselves.

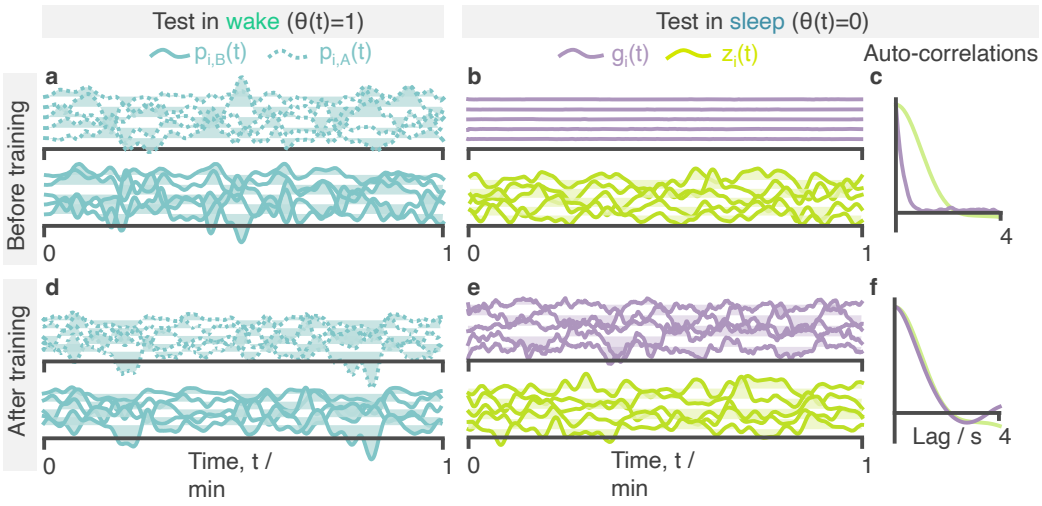

Figure D.1: Extended results from the artificial latent learning task. **a** Basal and apical voltages in the sensory layer before learning during a one minute sample in wake mode. **b** Samples of activity in the hidden layer and true latents before training during a one minute sample in sleep mode. **c** Autocorrelations, averaged over the units, for activity in panel b. **d**,**e** & **f** As in a, b & c but after training.

D.5 Path Integration Task: Implementation Details

An agent randomly moves around a 1 m 1D circular track. The trajectory, $x(t)$, is sampled using the RatInABox (George et al. 2024) simulation package. This means that velocity is modelled as an Ornstein-Uhlenbeck process (see eq. (D.8)) with a decorrelation timescale of $\tau = 0.7$ seconds and a standard deviation of $\sigma = 0.5$ ms^{-1}. There are $N_p = N_g = 100$ neurons in both

layers. The HPC dendritic activation function is linear ($\sigma_{p_A}(x) = x$) whilst both MEC dendritic compartments have rectified tanh activation functions ($\sigma_{g_B}(x) = \sigma_{g_A}(x) = \max(0, \tanh(x))$). Note the choice of activation function means MEC neurons have firing rate $\mathcal{O}(1\text{ Hz})$. All learning rates are set to $\eta = 0.01$, the discretisation timestep was $dt = 25$ ms and only $\mathbf{p}_A$ & $\mathbf{g}_B$ have learnable biases.

I model $N_i = N_p = 100$ inputs which are tuned to the position of the agent according to the following Gaussian tuning curves (these roughly model place cells):

$$[\mathbf{Œ}(t)]_i = \exp\left[-\frac{(x(t) - x_i)}{2\sigma^2}\right]. \tag{D.15}$$

where x_i are centres of the Gaussians evenly spaced along the track. These then linearly drive the basal dendritic compartments of the sensory neurons:

$$\mathbf{p}_B(t) = \mathbf{B}\phi(x(t)) \tag{D.16}$$

where, in the results shown in the main text, $\mathbf{B}_{ij} = \delta_{ij}$ is the identity matrix such that each sensory neuron inherits a unimodal-tuning curve from one and only one of the inputs, i.e. what was stated in eq. (4.8). I show in Figure D.2 that this choice is not particularly critical and the network can learn to perform path integration with random sensory drive ($[\mathbf{B}]_{ij} \sim \mathcal{N}(0, 1/\sqrt{N_p})$).

Velocity inputs are connected as follows: two neurons encode the rectified leftward and rightward velocity of the agent, normalised by the standard deviation σ. Note, this means they have firing rates $\mathcal{O}(1\text{ Hz})$.

$$\begin{aligned} v_L(t) &= \max(0, -\dot{x}(t)/\sigma) \\ v_R(t) &= \max(0, \dot{x}(t)/\sigma) \end{aligned} \tag{D.17}$$

Two sets of conjunctive cells ($N_g = 100$ in each set) sum inputs from the left

and right velocity neurons and the hidden units as follows:

$$[\mathbf{g}_{v_L}(t)]_i = \sigma_{gv}\Big(v_L(t) - v_R(t) + \sum_j [\mathbf{w}^{g_{vL}}]_{ij}[\mathbf{g}(t)]_j\Big)$$
$$[\mathbf{g}_{v_R}(t)]_i = \sigma_{gv}\Big(v_R(t) - v_L(t) + \sum_j [\mathbf{w}^{g_{vR}}]_{ij}[\mathbf{g}(t)]_j\Big) \tag{D.18}$$

where $\sigma_{gv}(x) = \max(0, x-1)$ is a ReLU function thresholded at $x = 1$. In the main text I set $[\mathbf{w}^{g_{vL}}]_{ij} = [\mathbf{w}^{g_{vR}}]_{ij} = \delta_{ij}$ so each conjunctive cell is connected to one and only one hidden unit (something I relax in Figure D.2c). The consequence of this connectivity is that a $\mathbf{g}_{v_L}$ neuron is above threshold (and therefore active) if and only if the agent is moving to the left*and* the hidden unit it is connected to is active. Rightward motion silences $\mathbf{g}_{v_L}$ neurons. Similarly, a $\mathbf{g}_{v_R}$ neurons is active if and only if the agent is moving to the right and the hidden unit it is connected to is active. This conjunctive, logic-AND-gate-like tuning to both MEC and velocity is why these neurons are called "conjunctive" cells.

To order the MEC neurons after learning, and thus reveal the ring attractor, I calculate their receptive fields as a function of agent position, $\mathbf{g}(x)$, as though the network is in inference mode (so top-down recurrent connections and drive from the conjunctive cells do not play a role). Then I permute the ordering $i' \leftarrow i$ such that the maxima of the receptive fields move from left to right along the track as the neuron count increases, $\arg\max_x [\mathbf{g}(x)]_{j'} > \arg\max_x [\mathbf{g}(x)]_{i'} \forall i', j' > i'$. The effect of this ordering procedure is shown in Figure D.2a (left hand side, top two panels).

Figure D.2a repeats the same path integration test as was shown in the main text Figure 4.3 except now I additionally visualise the receptive fields of HPC and MEC (after learning) and show timeseries of both HPC and MEC neurons during the test. Once MEC neurons are reordered by their maxima the ring attractor activity bump can be seen moving up at down the manifold of neurons, even after the sensory lesion. Note again how some MEC neurons have "died" and do not engage in the ring attractor dynamics, forcing the ring

attractor manifold to live on the remaining subset of MEC neurons.

D.5.1 Position Decoding

To quantify the performance of path integration I train a decoder to estimate agent position directly from the HPC population vector. The decoder is trained on position and activity data from the final 10 minutes of training, after learning had plateaued. The decoder I use is a Gaussian process regressor with a squared exponential kernel, the length scale of which is optimised during fitting. The decoder works well as can be seen in the path integration plots where, before the sensory lesion, the decoded position correctly and accurately tracks the true position.

D.5.2 Robustness to Weight Initialisations, Plasticity Lesions and Noise

Since a central claim of this chapter is that the network can learn, *from random initialisations*, the correct connectivity required to perform path integration, it is important to question where and why weights in the model are not randomly initialised and plastic.

Sensory weights The weights from the Gaussian tuned inputs to the HPC sensory neurons, $\mathbf{B}$ in eq. (D.16), must be non-plastic to prevent the network from rapidly converging on a trivial solution where all input weights fall to zero killing all activity in the network and trivially minimising the local prediction errors. They do not, however, need to be the identity function as I chose. Figure D.2b repeats the standard path integration experiment but with a network where $[\mathbf{B}]_{ij} \sim \mathcal{N}(0, 1/\sqrt{N_p})$, path integration is still learned without any problem. Ultimately this is not particularly surprising since the mapping from the spatially-tuned sensory inputs, ϕ, to the ring attractor in the original formulation was already mixed once by the randomly initialised weights from HPC to MEC ($\mathbf{w}_{g_B}$). This just adds one additional layer of mixing.

MEC to conjunctive cells I show in Figure D.2c, that path integration is still learned even when the MEC to conjunctive cell weights are initialised randomly,

$[\mathbf{w}_{g_{vL}}]_{ij} \sim \mathcal{N}(0, 1/\sqrt{N_g})$, $[\mathbf{w}_{g_{vR}}]_{ij} \sim \mathcal{N}(0, 1/\sqrt{N_g})$. I leave it to future work to investigate this result more thoroughly, and it is a notable relaxation on assumptions made in previous models (Vafidis et al. 2022; Burak et al. 2009) that fine-tuned connectivity from MEC to the conjunctive cells is assumed a priori for path integration (connectivity which would presumably have to be genetically encoded, which seems unlikely). I suspect part of the reason the path integration is robust with respect to the setting of these weights is down to the ability for MEC to construct its own inputs from HPC. This might mean the exact form of the activity bump inside the ring attractor can be tailored to fit the specific connectivity to the conjunctive cells – which is perhaps randomly determined during development – in a particular network.

Plasticity lesions Path integration, as explored in section 4.3.2, requires fine tuning the recurrent weights in the hidden layer (w_{g_A}) and consequently fails when this plasticity is turned off (Figure D.3a). Intriguingly however, I find that path integration does not *strictly* require plasticity between HPC and MEC (as shown in Figure D.3b, echoing results in (Vafidis et al. 2022)). However, when such plasticity is removed, the apical input to HPC coming from MEC is unmatched to the sensory input HPC receives from the environment. As such, any downstream system reading out position from the HPC code would only be able to do so during sleep or wake and not both. This is somewhat restrictive for a system hoping to use the hippocampal formation for online inference and planning. Hence, a primary role of interlayer plasticity between HPC and MEC in the model is to "translate" the environment-agnostic MEC code into the the environment-specific HPC code. This idea is discussed further in section 4.3.3.

D.6 Remapping Task: Implementation Details

To investigate remapping, the network is first trained to path integrate as described in the main text. The only difference is that I fix the weights from HPC to MEC to the identity matrix ($[\mathbf{w}_{g_B}]_{ij} = \delta_{ij}$ and $\eta = 0$ on these weights) during this phase of training, this results in MEC neurons with receptive fields

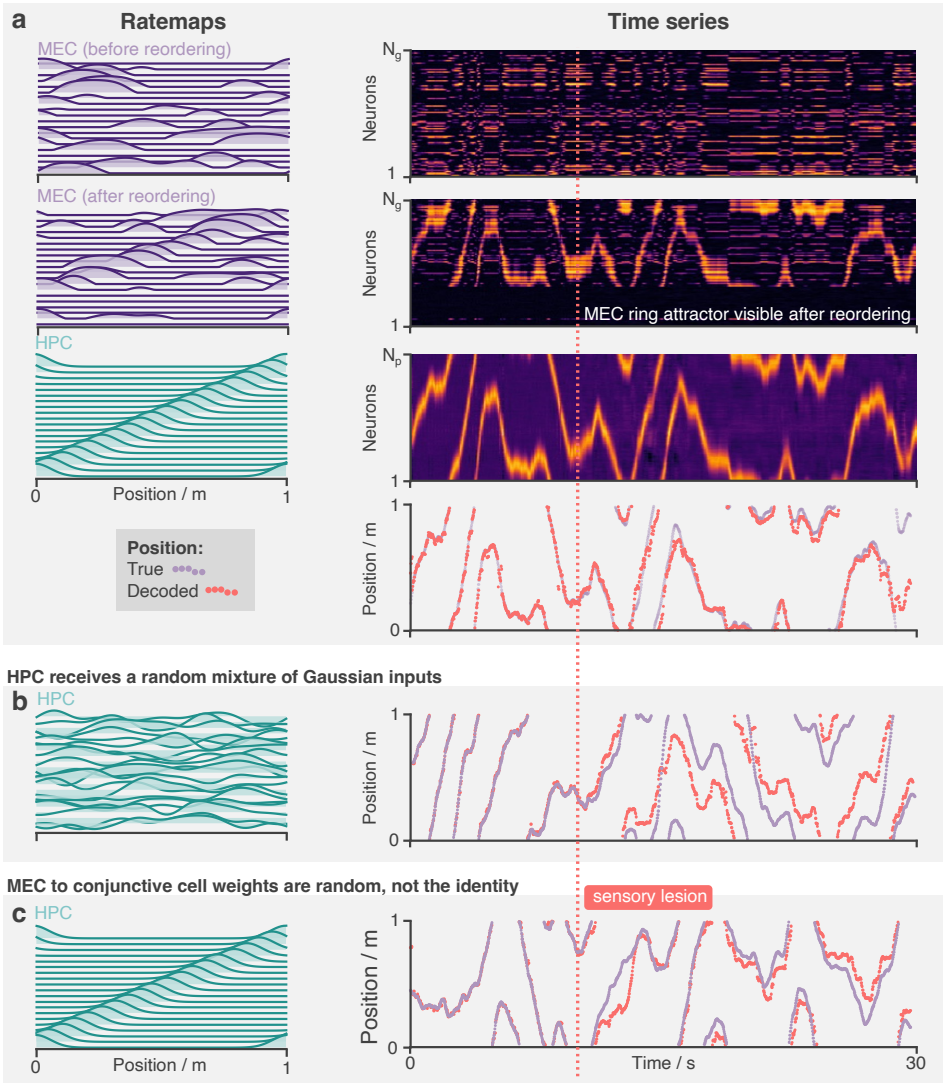

Figure D.2: Path integration is performed by a ring attractor in MEC revealed once the neurons are reordered by receptive field peak position. The network learns to path integrate robustly, regardless of the choice of random initialisations. **a** The same path integration test as in the main text is performed here: The top three rows show receptive fields (left) and timeseries activity (right) for the MEC (top two) and HPC layers (third) layers. MEC receptive fields and activity at first appears random. It is only after reordering the neurons by the peak position of their receptive fields that I see the ring attractor manifold. The bottom row shows the decoded position (red) and the true position (purple), demonstrating accurate path integration. **b** Like panel a except, instead of unimodal Gaussian inputs, the HPC neurons receive a random-sum-of-Gaussian inputs. Nonetheless the network still learns to path integrate (right). **c** Like panel a – with HPC neurons returned to their original Gaussian receptive fields – except in this experiment the hidden units (MEC, $\mathbf{g}$) are connected to the conjunctive cells randomly, not one-to-one. The network still learns to path integrate.

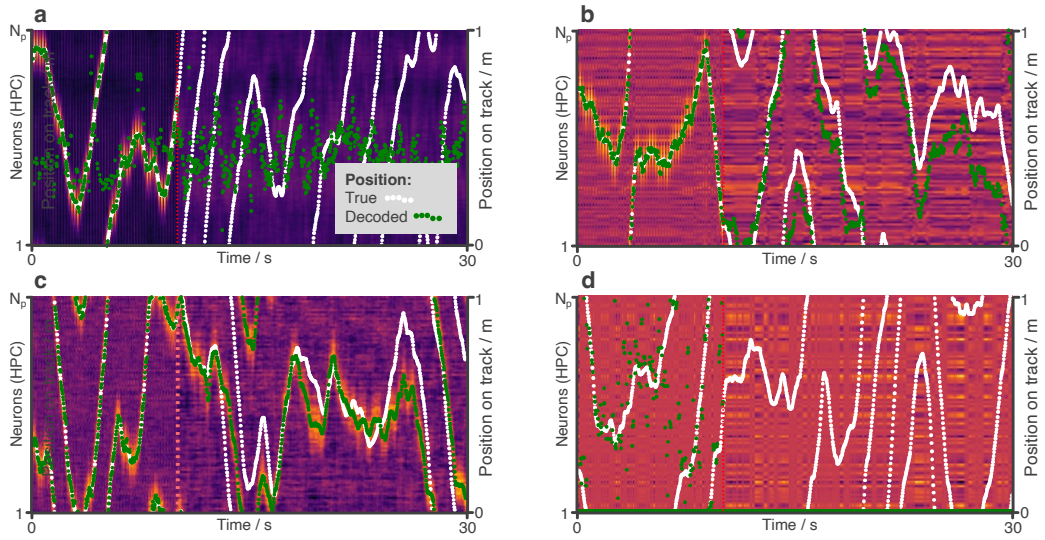

Figure D.3: Network response to removal of plasticity and additional noise. The standard path integration experiment is performed and hippocampal activity (as well as true and decoded position) is shown in four modified conditions. **a** Plasticity on the recurrent synapses (w_{g_A}) is turned off and the network no longer learns to path integrate. **b** Plasticity on all weights between HPC and MEC (w_{g_B} & w_{p_A}) is turned off. The network still learns to path integrate but inputs to HPC from MEC are not matched to those from the sensory input. **c** Synaptic noise on all synapses is increased by a factor of 10. The bump attractor is now noisier than Figure 4.3e but path integration is still accurate. **d** Synaptic noise on all synapses is increased by a factor of 100 at which point learning fails.

equal to those of the HPC neurons (except also passed through a rectified-tanh activation function), Figure 4.4b left column.

In the second phase I begin by randomly permuting the centres of the Gaussian sensory inputs in eq. (D.15). This "sensory shuffle" simulates the sort of hippocampal remapping event which typically occurs when an agent enters into a new environment. The activations of all neuronal layers are reset to zero. A second phase of learning then begins, this time only the weights from HPC to MEC ($\mathbf{w}_{g_B}$) and from MEC to HPC ($\mathbf{w}_{p_A}$) are plastic ($\eta = 0.01$) while the recurrent weights within MEC and the weights from the conjunctive cells to MEC (collectively, $\mathbf{w}_{g_A}$) are frozen ($\eta = 0$).

I found that MEC neurons regroup after the shuffle, reestablishing the pairwise correlational structure they had before remapping with, perhaps, a phase shift (Figure 4.4b). Once the ring attractor manifold has reappeared in

this way the ability to path integrate returns (Figure 4.4c). I find these results are clearest when $\mathbf{w}_{g_B}$ was fixed to the identity during the initial learning phase as described above. Although I don't investigate this finding thoroughly I suspect it is because the network has an easier time learning the ring attractor since the MEC inputs are already unimodal. With the identity mapping, a tidy activity bump already on the MEC cells before the rest of the ring attractor connectivity is learned, providing a good starting point. This matches the standard set up for studies of path integration in, for example, Vafidis et al. (2022)). This, perhaps, leads to a ring attractor which is more deeply embedded into the MEC recurrent connectivity structure and which can therefore more easily reestablish itself after a remapping. Nonetheless I discover that MEC is able to relearn a significant portion of the bump attractor structure during the second phase of learning even when this was not the case and $\mathbf{w}_{g_B}$ was randomly initialised ($\mathbf{w}_{g_B} \sim \mathcal{N}(0, 1/N_p)$) and plastic during the initial learning, this is shown in Figure D.4. Note how, in contrast to the receptive field shown in Figure 4.4b, the MEC neurons are now multimodal and additional bands of correlational structure (in addition to a global phase shift) appear after relearning. I leave it to future work to investigate this further.

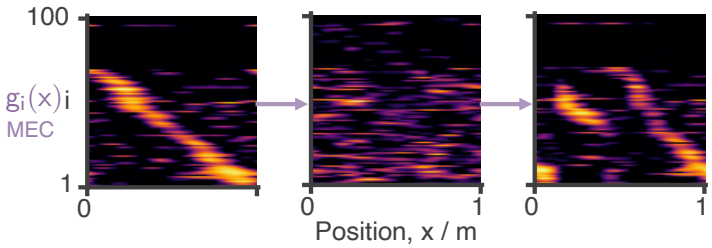

Figure D.4: Regrouping of the MEC neurons after sensory remapping but relaxing the constraint that HPC to MEC weights are fixed to the identity matrix during initial learning. This results in MEC neurons with multimodal receptive fields and more complex regrouping dynamics after remapping.

Appendix E

Appendix to Chapter 5

E.1 Code Availability

Code and demos for the SIMPL method are available at `https://github.com/TomGeorge1234/simpl`.

E.2 Theoretical Background: EM and State-Space Models

E.2.1 Expectation Maximization

Expectation Maximization (EM, Dempster et al. (1977)) is a widely used paradigm to perform statistical estimation in latent variable models. The goal of EM is to maximise the *Free Energy*, a lower bound on the log-likelihood $\log p(\mathbf{s};\mathbf{f})$ of the data, given by (following the notations of section 5.2.1):

$$\mathcal{F}(\mathbf{f}, q) \coloneqq \mathbb{E}_{q(\mathbf{x})}[\log p(\mathbf{x}, \mathbf{s}\,;\mathbf{f})] - \mathbb{E}_{q(\mathbf{x})}[\log q(\mathbf{x})] \leq \log p(\mathbf{s};\mathbf{f}), \tag{E.1}$$

where q is some probability distribution on the latent variable $\mathbf{x}$. Importantly $\mathcal{F}$ is maximised, and the lower bound becomes "tight", at $q^\star \coloneqq p(\mathbf{x}|\mathbf{s}\,;\mathbf{f})$, i.e. the posterior distribution of the latent variable given $\mathbf{s}$ and $\mathbf{f}$. Moreover, for a fixed q, the only $\mathbf{f}$-dependent term in $\mathcal{F}$ is $\mathbb{E}_{q(\mathbf{x})}[\log p(\mathbf{x}, \mathbf{s}\,;\mathbf{f})]$. To maximise $\mathcal{F}(\mathbf{f}, q)$ — and thus also increase the log-likelihood — EM produces a sequence $(\mathbf{f}^{(e)})_{e\geq 0}$ of parameters $\mathbf{f}^{(e)}$ by invoking, at each step (or "epoch") e, two well

known subroutines:

- **E-step**: Define $q^{(e)} \coloneqq p(\mathbf{x}|\mathbf{s}\,;\mathbf{f}^{(e-1)})$; compute $\mathcal{F} \longmapsto \mathbb{E}_{q^{(e)}}[\log p(\mathbf{x},\mathbf{s}\,;\mathbf{f})]$
- **M-step**: Compute $\mathbf{f}^{(e)} \coloneqq \arg\max_{\mathbf{f}} \mathcal{F}(\mathbf{f}, q^{(e)})$

$$= \arg\max_{\mathbf{f}} \mathbb{E}_{q^{(e)}}[\log p(\mathbf{x},\mathbf{s}\,;\mathbf{f})]$$

with the property that $\log p(\mathbf{s};\mathbf{f}^{(e)}) \geq \log p(\mathbf{s};\mathbf{f}^{(e-1)})$ for all e, grounding the use of EM to maximise the likelihood of the data. In this context it is important to note that, due to a Gaussianity assumption, calculating the expectation in the E-step requires estimating the mean (and variance) of the posterior $p(\mathbf{x}|\mathbf{s}\,;\mathbf{f}^{(e-1)})$ which can be treated as a point estimate of the latent trajectory, i.e. a "decoding" of the latent from the spikes. Thus, in the context of neural data, EM offers a framework to both estimate intensity functions via maximum likelihood, and also to decode the variable encoded by the neurons.

Impossibility of Exact EM for Gaussian-Modulated Poisson Processes The *E*-step of the EM algorithm requires computing a function *defined* as an expectation with respect to $p(\mathbf{x}|\mathbf{s}\,;\mathbf{f}^{(e-1)})$. In the case of Hidden Markov Models, such expectations are intractable to compute in closed form, unless the latent variable $\mathbf{x}$ is discrete (i.e. numerical estimation), or both the transition and the emission probabilities are Gaussian (with mean and variance depending linearly on $\mathbf{x}$, (Rauch et al. 1965)). In this particular case, exact inference in the model described in section 5.2.1 is impossible because the emission probabilities are Poisson with mean given by a non-linear function of $\mathbf{x}$ via each neurons tuning curve. In order to perform statistical inference for this spike train model — and avoid resorting to numerical estimation which is computationally expensive — SIMPL makes a set of approximations that are detailed below. At a high level the goal is to convert the non-linear, non-Gaussian spiking observations, into a variable that is linear and Gaussian with respect to the latent, thus EM can be performed exactly using a Kalman smoother.

E.2.2 Linear Gaussian State Space Models and Kalman Smoothing

Linear Gaussian State Space Models (LGSSM) are dynamical systems of the form:

$$\begin{aligned} \mathbf{z}_{t+1} &= F_t \mathbf{z}_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0_d, Q_t) \\ \mathbf{x}_t &= H_t \mathbf{z}_t + \delta_t, \quad \delta_t \sim \mathcal{N}(0_m, R_t). \end{aligned} \tag{E.2}$$

where $\mathbf{z} \in \mathbb{R}^d$, $\mathbf{x} \in \mathbb{R}^m$, $F_t, Q_t \in \mathbb{R}^{d \times d}$, $H_t \in \mathbb{R}^{p \times d}$ and $R_t \in \mathbb{R}^{m \times m}$. LGSSMs can be used as latent variable models given some observed data $\mathbf{x}$, where $\mathbf{z}$ is treated as a latent variable. While these models are limited in their expressiveness, their benefits are that inference (here, the "E-steps") can be done very efficiently: not only is the posterior $p(\mathbf{z}_1, \ldots, \mathbf{z}_T | \mathbf{x}_1, \ldots, \mathbf{x}_T)$ a Gaussian distribution (of dimension Td), but all of its marginals and pairwise marginals $p(\mathbf{z}_t | \mathbf{x}_1, \ldots, \mathbf{x}_T), p(\mathbf{z}_t, \mathbf{z}_{t+1} | \mathbf{x}_1, \ldots, \mathbf{x}_T)$ (crucially, the only distributions needed for learning the parameters of LGSSM via EM) can be computed jointly in $\mathcal{O}(T)$ time using an efficient technique known as Kalman Smoothing (Kalman 1960; Rauch et al. 1965).

Such a scaling contrasts with naive numerical binning-based alternatives for inference in continuous, non-Gaussian State Space Models, which require maintaining an estimate of each bin — a vector of size n (no. bins) where n grows *exponentially* with the dimension of the latent space, as used in e.g. Denovellis et al. (2021). Instead, for LGSSMs, the Gaussianity means only the mean and covariance of the marginal posterior distributions — of size d and d^2 respectively — need to be stored. This is not memory intensive and, perhaps more importantly, the Kalman Filter proceeds to compute them in a combined $\mathcal{O}(T)$ time. In the experiments performed here, it was found that the cost of the Kalman Filter was negligible relative to the kernel evaluations that are the main computational bottleneck of SIMPL.

From here onwards, $\mathbf{x}$ will be used to denote the latent variable in the LGSSM, and $\widehat{\mathbf{x}}$ or $\mathbf{s}$ for observations.

E.3 SIMPL as Approximate EM

E.3.1 MLE-Based Approximate E-Step

Instead of $q^{(e)} = p(\mathbf{x}|\mathbf{s}\,;\mathbf{f}^{(e-1)})$, SIMPL computes an approximation to $q^{(e)} \approx \widehat{q}^{(e)} = p(\mathbf{x}|\widehat{\mathbf{x}}\,;\mathbf{f}^{(e-1)})$ where $\widehat{\mathbf{x}}$ is the Maximum Likelihood Estimate (MLE) of $\mathbf{x}$ given the observations $\mathbf{s}$ and the current tuning curves $\mathbf{f}^{(e-1)}$ defined as:

$$\begin{aligned} \widehat{\mathbf{x}} &= \arg\max_{\mathbf{x}} \log p(\mathbf{s}|\mathbf{x}\,;\mathbf{f}^{(e-1)}) = \arg\max_{\mathbf{x}} \sum_{t=1}^{T}\sum_{i=1}^{N} \log p(s_{ti}|\mathbf{x}_t\,;\mathbf{f}^{(e-1)}) \\ \implies \widehat{\mathbf{x}}_t &= \arg\max_{\mathbf{x}_t} \sum_{i=1}^{N} \log p(s_{ti}|\mathbf{x}_t\,;\mathbf{f}^{(e-1)}). \end{aligned} \tag{E.3}$$

As defined, computing the MLE returns a point estimate of the *true* trajectory that led to the observed spike train $\mathbf{s}$, however a posterior is sought. In particular, MLE does not use the prior knowledge encoded by $p(\mathbf{x})$.

To find the approximate posterior it is noted that, as a function of $\mathbf{s}$, $\widehat{\mathbf{x}}$ is itself a random variable. In the many neurons limit, and under certain regularity assumptions, the distribution of this random variable converges to a Gaussian, a fact known as *asymptotic normality*. In other words; though $\mathbf{s}$ (conditioned on $\mathbf{x}$) is a non-Gaussian random variable, $\widehat{\mathbf{x}}$ (a deterministic function of $\mathbf{s}$) is approximately Gaussian in the many neurons limit and thus satisfied the conditions of the LGSSM.

I restate a formal statement of this asymptotic normality result in the case of independent, but non-identically distributed observations[1] originally established in (Bradley et al. 1962), and reformulated using the notations of the model at hand. For simplicity, the case where only P distinct intensity functions $\mathbf{f}_1, \ldots, \mathbf{f}_P$ exist will be considered, although versions of this result exist without this assumption.

Theorem E.3.1 (Asymptotic Normality of the MLE)**.** *Let* $\mathbf{x}_t^\star \in \mathbb{R}^d$*. Let* $\mathbf{s} = (s_{1t}, \ldots, s_{Nt})$ *be independent random variables with probability densities* $p(s_{ti}|\mathbf{x}_t^\star; \mathbf{f}_{t(i)})$*, where* $t(i) \in \{1, \ldots, P\}$ *is the index of the intensity function*

[1] The i.i.d case was established in (Fisher 1925)

$f_{t(i)}$ that generated the spike train s_{ti}. For $p \in 1, \ldots, P$, denote n_p the number of times the intensity function f_p appeared in the sequence $\mathbf{f}_{t(i)}$. Assume that the MLE $\widehat{\mathbf{x}}_t$ exists and it is unique. Then, under mild regularity conditions, we have:

$$\sqrt{N}\left(\widehat{\mathbf{x}}_t - \mathbf{x}_t^\star\right) \xrightarrow[N\to\infty]{\mathrm{d}} \mathcal{N}(0, \mathcal{I}(\mathbf{x}_t^\star)^{-1})$$

where $\mathcal{I}(\mathbf{x}_t^\star) \coloneqq \sum_{p=1}^{P} \mu_p \mathbb{E}_{p(\mathbf{s}_t;\mathbf{f}_p)} \mathrm{Hess}(\log p(\mathbf{s}_t|\mathbf{x}_t^\star;\mathbf{f}_p))$ is the Fisher Information matrix of the model at $\mathbf{x}_t^\star$, $\xrightarrow{d}$ means convergence in distribution, and it is defined that $\mu_p \coloneqq \lim_{N\to\infty} \frac{n_p}{N}$.

The asymptotic Gaussianity of the MLE in the many neurons limit suggests performing approximate inference in a surrogate Hidden Markov Model, with the same transition probabilities $p(\mathbf{x}_{t+1}|\mathbf{x}_t)$ as the original ones, but where the observations $\mathbf{s}$ are replaced by $\widehat{\mathbf{x}}$. Leveraging Theorem E.3.1, SIMPL approximates the emission probabilities $p(\widehat{\mathbf{x}}_t|\mathbf{x}_t)$ by the Gaussian distribution $\mathcal{N}(\mathbf{x}_t, \mathbf{\Sigma}_t)$, where $\mathbf{\Sigma}_t \coloneqq (N\mathcal{I}(\widehat{\mathbf{x}}_t))^{-1} \approx (N\mathcal{I}(\mathbf{x}_t))^{-1}$, the Fisher information of the spikes.

Temporarily ignoring the $\mathbf{x}_t$-dependence of the covariance matrices $\mathbf{\Sigma}_t$ and treating them as deterministic (discussed below), the variables $(\mathbf{x}_t, \widehat{\mathbf{x}}_t)$ then form the following latent variable system with hidden variables $\mathbf{x}_t$ and observed variables $\widehat{\mathbf{x}}_t$ given by:

$$\begin{aligned} \mathbf{x}_{t+1} \mid \mathbf{x}_t &\sim \mathcal{N}(\mathbf{x}_t, \sigma_v^2 \mathbf{I}), \\ \widehat{\mathbf{x}}_t \mid \mathbf{x}_t &\sim \mathcal{N}(\mathbf{x}_t, \mathbf{\Sigma}_t) \end{aligned} \tag{E.4}$$

This model is precisely an instance of Linear Gaussian State Space Models defined in eq. (E.2) and the four matrices set to:

$$\begin{aligned} F_t &= \mathbf{I} && \text{(constant)} \\ H_t &= \mathbf{I} && \text{(constant)} \\ Q_t &= \sigma_v^2 \mathbf{I} && \text{(constant)} \\ R &= \mathbf{\Sigma}_t && \text{(time-varying).} \end{aligned} \tag{E.5}$$

This correspondence allows SIMPL to compute an approximation of the marginal posterior distributions $p(\mathbf{x}_t|\mathbf{s}) \approx p(\mathbf{x}_t|\widehat{\mathbf{x}})$ using Kalman Smoothing (Kalman 1960; Rauch et al. 1965). Importantly, the MLE estimates $\widehat{\mathbf{x}}_t$ can be obtained in parallel for all t; the only sequential procedure remaining being the Kalman Smoothing step. The trajectory $\widehat{\mathbf{x}}^{(e)}$ of SIMPL's E-step is then set to the mean of $\widehat{q}^{(e)}$.

E.3.2 Spike Smoothing: A Generalized M-Step

SIMPL's M-step is computationally cheap and interpretable. However, it differs from the M step of the EM algorithm, which is recalled to be given by:

$$\mathbf{f}^{(e),\mathrm{EM}} := \arg\max_{\mathbf{f}} \mathcal{F}(\mathbf{f}, q^{(e)}) = \arg\max_{\mathbf{f}} \mathbb{E}_{q^{(e)}}[\log p(\mathbf{x}, \mathbf{s}\,; \mathbf{f})] \tag{E.6}$$

Below, I reconcile the two approaches by showing that SIMPL M-step can be seen as instances of a more general "model fitting" M-step. To see why, note the objective function of a standard M-step equals (up to a constant in $\mathbf{f}$) the negative KL divergence between the joint distribution[2] $\widehat{q}^{(e)}(\mathbf{x}|\mathbf{s})p(\mathbf{s})$ (observable through samples) and the model $p(\mathbf{s}, \mathbf{x}; \mathbf{f})$. Thus, a standard M-step can be understood as minimizing this KL divergence approximately, by replacing the expectation over $p(\mathbf{s})$ by an empirical average over the true data $\mathbf{s}$, an approximation which is asymptotically consistent in the large number of time steps limit under suitable ergodicity conditions (Billingsley 1961).

Similarly, SIMPL's M-Step also fits the model $p(\mathbf{s}, \mathbf{x}, \mathbf{f})$ to the "data" distribution $\widehat{q}^{(e)}(\mathbf{x}|\mathbf{s})p(\mathbf{s})$. However, instead of doing so by minimizing the KL divergence between the data and the model, it does so using a kernel-based estimate [3]. Thus, Both M-steps can be understood as having the same goal, simply differing in their solution to solve it. In that sense, SIMPL's M-step is indeed a generalized M-step.

[2] I denote $q^k(x)$ by $q^k(x|s)$ to highlight the dependence between x and s.

[3] Additionally, it replaces the expectation over $\widehat{q}^{(e)}(\mathbf{x}|\mathbf{s})$ by a one-sample estimate of it through $\tilde{\mathbf{x}}$

E.4 Efficient Implementation and Algorithmic Details

Some implementation details that were important to maximise the computational efficiency of the method are provided below.

E.4.1 Maximizing Computational Efficiency

Computational Bottlenecks

A single evaluation of the log-likelihood $\log p(\mathbf{s}_t|\mathbf{x}_t)$ requires evaluating the kernel-based rate map estimates given in eq. (E.6). This takes $\mathcal{O}(T)$ time since it involves a sum across all timesteps. Moreover, this calculation will be repeated itself T-times for each step of the Kalman smoother in order to (1) compute the MLEs $\widehat{\mathbf{x}}_t$ (that naively require gradient ascent on $\log p(\mathbf{s}_t|\mathbf{x}_t)$) and (2) evaluate the MLE variance $\mathbf{\Sigma}_t \coloneqq (N\mathcal{I}(\widehat{\mathbf{x}}_t))^{-1} = (N\mathbf{H}_x(\log p(\mathbf{s}|\widehat{\mathbf{x}}_t))(\widehat{\mathbf{x}}_t))^{-1}$. All in all, an exact implementation of SIMPL's E-step would have quadratic $\mathcal{O}(T^2)$ time complexity, which would be prohibitively slow for long datasets. Moreover, the second-order differentiation needed to compute $\mathcal{I}(\widehat{\mathbf{x}}_t)$ is also computationally expensive (formally, it introduces a large constant factor in front of the $O(T^2)$ term). In the next sections, additional approximations are described that allow SIMPL to estimate the MLE and its variance in $\mathcal{O}(T)$ time and without differentiating the rate maps.

Linear-Time MLE estimation

Naive gradient-based solution The naive way to calculate the MLE $\widehat{\mathbf{x}}_t$ is to evaluate all N tuning curves (recall each evaluation costs $\mathcal{O}(T)$) for some location $\mathbf{x}$, use these to establish the log-likelihood $\log p(\mathbf{s}_t|\mathbf{x})$, calculate the gradient of this log-likelihood with respect to $\mathbf{x}$, and then take, for example, k gradient descent steps to find the MLE. This process is repeated for each timestep t in the Kalman smoother, leading to a quadratic time complexity of $\mathcal{O}(kNT^2)$.

SIMPL's approach To compute the MLE in linear time SIMPL bypasses the need to recalculate the tuning curves at each time step by, instead, binning them onto a discretized grid of points once at the start of each iteration. Formally SIMPL computes n evaluations the tuning curves $\tilde{\mathbf{f}} \coloneqq (\tilde{\mathbf{f}}_1, \dots, \tilde{\mathbf{f}}_n) \coloneqq (\mathbf{f}(\mathbf{g}_1), \dots, \mathbf{f}(\mathbf{g}_n))$ on a grid of n points $\mathcal{G} = (\mathbf{g}_1, \dots, \mathbf{g}_n)$. This has time complexity $\mathcal{O}(NnT)$. I use a uniform rectangular grid of points (the smallest rectangle containing the full observed behavioural variable) with a spacing dx. For example, in a 1 m $\times$ 1 m environment with $dx = 0.02$ m, this would yield a grid of 50×50 points ($n = 2500$). Then, given $\tilde{\mathbf{f}}$, SIMPL then discretizes the log-likelihood functions $\log p(\mathbf{s}_t|\mathbf{x})$ over that same grid:

$$\begin{aligned} \tilde{l}_{it} \coloneqq \log p(\mathbf{s}_t|\mathbf{g}_i) &= \sum_{j=1}^{N} \log p(s_{tj}|\mathbf{g}_i) = \sum_{j=1}^{N} \log \frac{e^{-\tilde{f}_{ij}} \tilde{f}_{ij}^{s_{tj}}}{s_{tj}!} \\ &= -\sum_{j=1}^{N} \tilde{f}_{ij} + s_{tj} \log \tilde{f}_{ij} - \log s_{tj}! \end{aligned} \tag{E.7}$$

where it is noted that $\tilde{f}_{ij} \coloneqq [\tilde{\mathbf{f}}(\mathbf{g}_i)]_j$. Finally, given such evaluations, SIMPL sets its approximation of the MLE to be

$$\widehat{\mathbf{x}}_t \coloneqq \arg\max_{\mathbf{g} \in \mathcal{G}} \log p(\mathbf{s}_t|\mathbf{g}) = \arg\max_i \tilde{l}_{it} \tag{E.8}$$

This way of calculating the MLE has linear time complexity yielding an improvement for $n < kT$.

Linear-Time Derivative-Free MLE Variance Estimation

A similar strategy could be employed to also compute

$$\mathcal{I}(\widehat{\mathbf{x}}_t) \coloneqq -\mathbf{H}_x(\log p(\mathbf{s}_t|\widehat{\mathbf{x}}_t))(\widehat{\mathbf{x}}_t), \tag{E.9}$$

which appears in $\mathbf{\Sigma}_t$. Here $\mathbf{H}_x$ is the Hessian operator defined as $\mathbf{H}_x(f)(x) \coloneqq \nabla_x^2 f(x)$. To do so, one could compute the Hessian of the rate maps and their logarithm on that grid, from which any $\mathbf{H}_x(\log p(\mathbf{s}|\widehat{\mathbf{x}}_t))(\widehat{\mathbf{x}}_t)$ at the grid-point-based MLE obtained above can be evaluated as $\mathbf{H}_x(\log p(\mathbf{s}_t|\mathbf{g}_i))(\mathbf{g}_i) =$

$-\sum_{j=1}^{N} \mathbf{H}_x(f_j)(\mathbf{g}_i) + s_{tj}\mathbf{H}_x(\log f_j)(\mathbf{g}_i)$. This would be linear in time however, it was found that differentiating $\mathbf{f}$ could be very slow.

Instead SIMPL takes an entirely different approach and produces an estimation of $\mathbf{\Sigma}_t$ by instead *estimating the variance of the posterior distribution* $p(\mathbf{x}_t|\mathbf{s}_t) \propto p(\mathbf{x}_t)p(\mathbf{s}_t|\mathbf{x}_t) = p(\mathbf{s}_t, \mathbf{x}_t)$. The posterior variance and the MLE variance are expected to closely match, as discussed in our theoretical justification above. Moreover, as this posterior is available analytically up to the normalizing constant $p(\mathbf{s}_t)$, its variance can be approximately computed by binning $p(\mathbf{x}_t|\mathbf{s}_t)$ onto the same grid $\mathcal{G}$ introduced above, yielding the following fast estimator for $\mathbf{\Sigma}_t$.

$$\mathbf{\Sigma}_t \approx \operatorname{Cov} p(\mathbf{x}_t|\mathbf{s}_t) \approx \frac{\sum_i \tilde{p}_{it}(\mathbf{g}_i - \boldsymbol{\mu}_t)(\mathbf{g}_i - \boldsymbol{\mu}_t)^T}{\sum_i \tilde{p}_{it}}, \quad \boldsymbol{\mu}_t \coloneqq \frac{\sum_i \mathbf{g}_i \tilde{p}_{it}}{\sum_i \tilde{p}_{it}} \tag{E.10}$$

where $\tilde{p}_{it} \coloneqq \exp(\tilde{l}_{it}) = p(\mathbf{s}_t|\mathbf{g}_i)$. Intuitively, this is equivalent to fitting a multivariate Gaussian to the binned likelihood map. The covariance matrix of this Gaussian is then used as an approximation of the MLE variance. A theoretical argument justifying the validity of this formula is provided below.

Theoretical Justification Equation (E.10) is justified by the Bernstein Von Mises theorem, which states that the difference between the posterior distribution and the distribution of the MLE vanishes in the many neurons limit. I restate this theorem using the notations of this chapter, assuming a unique rate map, and without stating some of the required regularity assumptions for simplicity. I refer the reader to (Vaart 2000, Theorem 10.1, p.141–144) for the full version.

Theorem E.4.1 (Bernstein-von Mises)**.** *Let* $\mathbf{x}_t^\star \in \mathbb{R}^d$*. Let* $\mathbf{s}_t = (s_{1t}, \dots, s_{Nt})$ *be i.i.d random variables with probability density* $p(\mathbf{s}_t|\mathbf{x}_t^\star; \mathbf{f})$*. Assume that the MLE* $\widehat{\mathbf{x}}_t$ *exists and it is unique. Then, under mild regularity conditions, for any prior* p *on* $\mathbf{x}_t$*, we have:*

$$\|p(\mathbf{x}_t|\mathbf{s}_t) - \mathcal{N}(\widehat{\mathbf{x}}_t, (N\mathcal{I}(\mathbf{x}_t^\star))^{-1})\|_{\mathrm{TV}} \overset{p(\mathbf{s}_t)}{\underset{N\to\infty}{\to}} 0$$

where $\overset{p(\mathbf{s})}{\to}$ *denotes convergence in probability, and* $\|\cdot\|_{\mathrm{TV}}$ *denotes the Total Variation norm on bounded measures.*

From this theorem, I thus have that the (random) posterior distribution behaves (in total variation) as a Gaussian whose covariance matrix is precisely the asymptotic variance of the MLE. Note however that convergence in total variation does not a priori imply convergence of variances. Further work could examine under which assumptions such a convergence of variances may hold. In practice, I found that this approximation yielded a satisfying trade-off between performance and accuracy.

E.4.2 Iterative Linear Realignment of the Trajectories

To improve the identifiability properties and the numerical stability of SIMPL, I also transform the decoded latent trajectory at each iteration using a linear mapping which maximally aligns it with behaviour defined as $\mathbf{x}_t^{(e)} \leftarrow \mathbf{M}\mathbf{x}_t^{(e)} + \mathbf{c}$ where $\mathbf{M}, \mathbf{c} = \arg\min \sum_t \|\mathbf{x}_t^{(0)} - (\mathbf{M}\mathbf{x}_t^{(e)} + \mathbf{c})\|$. This approach ensures the scale, orientation and centre of the optimised latent trajectory are tied to behaviour, preventing accumulation of linear shifts/rotations across iterations and allowing the latent to be interpreted relative to, and in the same units as, behaviour. I suspect that performing this alignment on all iterates *after* the optimisation would yield similar results. Because the transformed latent necessarily has similar scale to the behaviour — which was used to set the size of the discretised environment — I can reuse the same discrete grid for the latent avoiding the need to rediscretize the environment at each iteration.

E.4.3 Hyperparameters Settings

SIMPL has two model hyperparameters:

- v: the diffusion rate for Kalman smoothing, which sets a prior over expected velocity of the latent variable. Units are in ms^{-1}.
- σ: the bandwidth of the kernel used in the M-step to smooth spikes. Units are in m.

Additionally there are some implementation-specific parameters:

- dx: the bin size for the variance estimation of the MLE. Units are in m.
- dt: the time step of the discretization of the latent variable. Units are in s.
- E: the number of iterations of the EM algorithm.

Finally, in all simulations I used a test fraction of 10% and held out 'speckled' data segments of length 1 second to evaluate the performance of the model. The value of these hyperparameters for the Artificial Grid Cell Dataset and the Real Hippocampal Dataset are provided in table E.1.

Table E.1: Hyperparameters settings for the SIMPL experiments

Dataset	v	σ	dx	dt	E
Artificial Grid Cell Dataset (fig. 5.2)	0.4 ms^{-1}	0.02 m	0.02 m	0.1 s	10
Real Hippocampal Dataset (fig. 5.3)	1.0 ms^{-1}	0.1 m	0.04 m	0.2 s	10
Motor task dataset (2D)[a] (fig. 5.4c,d)	1.0	0.1	0.02	0.05 s	10
Motor task dataset (4D) (fig. 5.4e)	1.5	0.09	0.1	0.05 s	10

[a] All behavioural variables (hand position and velocity) are normalised to lie between 0 and 1, so units here are arbitrary.

E.4.4 Synthetic Data Generation with the RatInABox Package

All synthetic grid cell data were generated using the RatInABox package (George et al. 2024). In this model, an agent moves through a 1 m by 1 m environment following a smooth, continuous random motion policy (details can be found in the original paper)under default parameters whereby the agent's

mean speed is 0.08 m/s. Specifically, the RatInABox model was used to generate the true latent trajectory, denoted as $\mathbf{x}^\star$. This trajectory was then "noised" to produce the trajectory used as the initial condition for SIMPL, denoted as $\mathbf{x}^{(0)}$, which represents the animal's measured position. The noise, or "discrepancy" vector $\Delta\mathbf{x} := \mathbf{x}^{(0)} - \mathbf{x}^\star$, was generated by sampling a velocity trajectory from a 2D Ornstein-Uhlenbeck process with zero mean and a coherence time scale of 3 seconds. This velocity trajectory was then integrated to obtain the discrepancy. Finally, the agent's behaviour $\mathbf{x}^{(0)}$ was additionally influenced by the same environmental forces implemented in the standard RatInABox model, i.e. the agent smoothly drifts away from walls to avoid crashing. The scale of the Ornstein-Uhlenbeck process was adjusted so that the mean discrepancy between the latent and observed trajectories was 20 cm in pen-space (i.e. away from walls).

Grid cells were modelled as the thresholded sum of three cosine plane waves (see Appendix A.2.3) with a width ratio—ratio of field width to inter-field distance—of 0.55. Each grid cell is assigned a wave direction θ_i, gridscale λ_i and 2D phase offset $\phi_i = [\phi_1, \phi_2]^\mathsf{T}$. Specifically, N=225 grid cells were divided into 3 modules of N=75 grid cells. Within each module all cells had the same grid scale (0.3, 0.5 and 0.8 m) and wave direction (0, 0.1 and 0.2 rad) but random phase offsets, approximately matching grid cells in the brain. Grid cell firing rates are all scaled to a maximum of 10 Hz.

Grid cells firing rates are determined by the latent position at time t, i.e. $f_i(t) = f^{GC}(\mathbf{x}_t^\star)$, from which spikes are sampled according to an inhomogeneous Poisson process $N_{spikes}(t, t+dt) \sim \text{Poi}(f_i(t) \cdot dt)$.

E.4.5 Test-Train Partitioning

To assess performance I partition the spike data matrix, $\mathbf{s}$, into testing and training sets, $\mathcal{S}_{\text{test}}, \mathcal{S}_{\text{train}}$. Inference is performed solely on the training set and I then track the log-likelihood of data in both sets (fig. 5.2d, left), e.g. $\ell^{(e)} = |\mathcal{S}_{\text{test}}|^{-1}_{\text{test}} \sum_{(i,t)\sim\mathcal{S}^{\text{test}}} \log p(s_{ti}|\mathbf{x}_t^{(e)}, \mathbf{f}_i^{(e)})$. This partitioning requires careful consideration: entire time intervals cannot be withheld for testing

without impairing the model's ability to infer the latent over this period. Likewise, entire neurons cannot be withheld without impairing the model's capacity to estimate their tuning curves. Instead, I adopt a speckled train-test mask previously used in latent variable modelling set-ups (Williams et al. 2020) which withholds for testing extended chunks of time bins arranged in an irregular "speckled" pattern across the data matrix (totalling 10% of the data).

E.4.6 Benchmarking Details

In section 5.3.5, I benchmarked SIMPL against four comparable methods on the synthetic grid cell datasets. For all techniques, tuning curves were visualised in the same way as for SIMPL: by extracting the latent trajectories after optimisation and using kernel smoothing to construct the rate maps, i.e., a single "M-step." Default parameters were used for all methods. CEBRA constrains its N-dimensional latent space to lie on an (N-1)-dimensional hypersphere. I disabled this constraint for the 2D grid cell dataset. For GPLVM, I used a variant that exploits induction points to improve scalability with the amount of data, performed a grid-search across the number induction points, and reported the best result. For GPDM, which does not feature induction points and thus has a cubic complexity with respect to T, training was restricted for this technique to 5 minutes of data (compared to 60 minutes for the other methods) to keep computation times under 1 hour,. For pi-VAE, I set the "task variable" u_t to the behaviour $x_t^{(0)}$, and the latent z_t to be the true position x_t. Additionally, I set the distribution $p(z_t|u_t)$ (e.g. the position given behavior mapping) to be a Gaussian distribution centered at $u_t = x_t^{(0)}$, and a fixed variance.

E.5 Supplementary Analyses and Robustness Tests

E.5.1 Discrete Latent Toy Model

Before testing SIMPL on a large temporally continuous dataset, a smaller dataset was constructed akin to a discrete two-alternative forced choice task (2AFC) (Figure E.1) — a widely studied decision–making paradigm (Platt et al. 1999; Bogacz et al. 2006; Znamenskiy et al. 2013; Lieder et al. 2019). The true latent states $\mathbf{x}_t^\star \in \{0, 1\}$ are binary and have no temporal structure (here subscript t indexes *trials* not time), analogous to a series of random "left" or "right" choices (fig. E.1b). This latent state is stochastically encoded by a population of neurons with random tuning curves giving the Bernoulli emission probabilities under each latent state:

$$f_i^\star(\mathbf{x}) = \begin{cases} f_{i0} \sim \mathcal{U}(0,1) & \mathbf{x} = 0, \\ f_{i1} \sim \mathcal{U}(0,1) & \mathbf{x} = 1, \end{cases} \tag{E.11}$$

$$\mathbf{x}_t^\star \sim \text{Bernoulli}(0.5) \quad \text{and} \quad s_{ti}|\mathbf{x}_t \sim \text{Bernoulli}(f_i^\star(\mathbf{x}_t^\star)). \tag{E.12}$$

Data is then sampled for $T = 50$ trials and $N = 15$ neurons as shown in fig. E.1. Initial conditions, $\mathbf{x}_t^{(0)}$, are generated from the true latent by randomly resampling a fraction of trials $\rho = 0.5$ (fig. E.1b). This partial resample represents an initial discrepancy between the behavioural measurement and the true internal state of the agent.

I perform inference on this dataset using a reduced version of the model (SIMPL-R). In the M-step, tuning curves were fitted by calculating the average activity of a neuron across each latent condition (e.g. $f_i^{(e)}(\mathbf{x}) = \sum_t s_{ti}\delta(\mathbf{x}_t^{(e)}, \mathbf{x}) / \sum_t \delta(\mathbf{x}_t^{(e)}, \mathbf{x})$, conceptually similar to kernel smoothing). For the E-step, each latent was the decoded according to the maximum likelihood estimate under the observed spikes and tuning curve estimates from the previous epoch: $\mathbf{x}_t^{(e+1)} = \arg\max_{\mathbf{x}} \sum_i \log p(s_{ti}|\mathbf{x}, f_i^{(e)})$ (there is no time dependence

between latents, thus no Kalman smoothing). This process was repeated for 5 epochs and, with high reliability, converged on the true latents after approximately two (fig. E.1c,d, distributions show repeat for 1000 randomly seeded datasets, dotted lines show ceiling performance on a model perfectly initialised with noiseless $\mathbf{x}^{(0)} = \mathbf{x}^\star$). I repeated this experiment for various values of ρ: latent recovery was almost perfect when ρ was small (i.e. when the initial conditions were close to the true latent), dropping off as ρ approached 1. At $\rho = 1$ when the conditions were *completely* random, the model was biased to recover a latent space that is either perfectly correlated or perfectly anti-correlated ("left" $\leftrightarrow$ "right") with the true latent (fig. E.1c, right), an isomorphic solution.

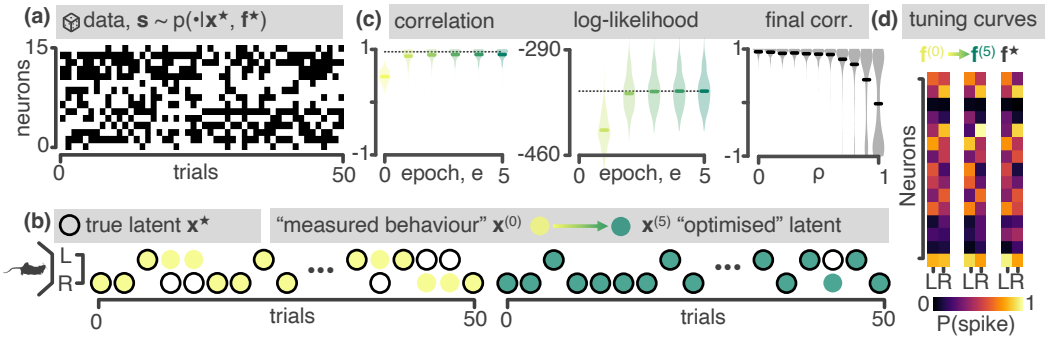

Figure E.1: A two-alternative forced choice task (2AFC) toy-model. **(a)** Data generation: Spikes are sampled from a simple generative model. For each of T=50 independent trials a random binary latent — analogous to a "left" or "right" choice — is encoded by a population of N=15 neurons with randomly initialised tuning curves. **(b)** Model performance: Starting from a noisy estimate (yellow) of the true latent (black) where a fraction $\rho = 0.5$ of trials are resampled, SIMPL-R recovers the true latent variables (green) with high accuracy. **(c)** *Left:* Correlation between $\mathbf{x}^{(e)}$ and $\mathbf{x}^\star$. *Middle:* Log-likelihood, $\log p(\mathbf{s}|\mathbf{x}^{(e)}, \mathbf{f}^{(e)})$. *Right:* Final correlation between $\mathbf{x}^{(5)}$ and $\mathbf{x}^\star$ as a function of initialization noise ρ. Violin plots show distributions over 1000 randomly seeded datasets, dotted lines show ceiling performance of a perfectly initialised model ($\mathbf{x}^{(0)} = \mathbf{x}^\star$) **(d)** Tuning curves.

E.5.2 Hyperparameter Sweep

I swept over the two hyperparameters v (the velocity prior) and σ (the kernel bandwidth) to assess how sensitive SIMPL is to these hyperparameters, as shown in Figure E.2. For this, the same synthetic grid cell dataset used in

fig. 5.2 was used. Notably, SIMPL's performance (measured in terms of the final error, see panel b) is relatively stable across a wide range of hyperparameters; kernel bandwidths between 0.1 cm and 5 cm and velocity priors between 0.2 m/s and 1 m/s all yield similar performance. Inspecting the tuning curves confirmed that kernel bandwidth has a significant effect on their appearance. Broader kernels give smoother tuning curves, eventually blurring the individual grid fields together, while narrower kernels give sharper tuning curves, eventually leading to overfitting where individual spikes are resolved.

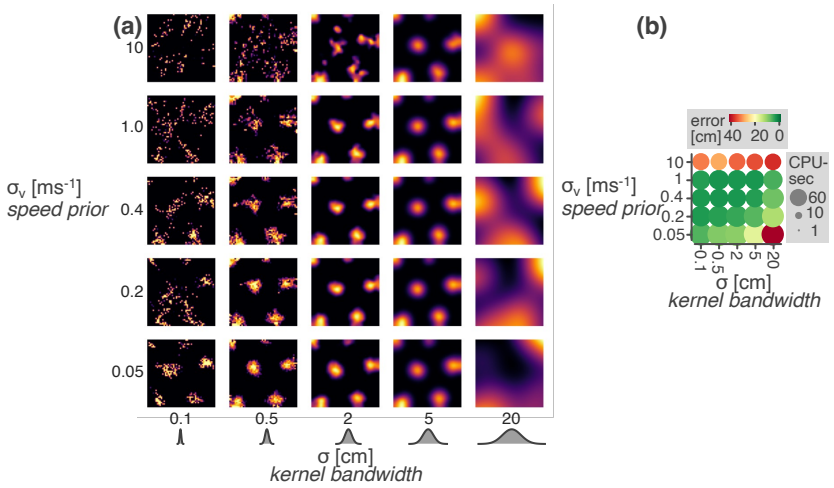

Figure E.2: Performance of SIMPL on the synthetic grid cell dataset as a function of the hyperparameters v (speed prior) and σ (kernel bandwidth). **(a)** Tuning curves. **(b)** Final error between the latent and ground truth (colour) and total compute time (size).

E.5.3 Non-Continuous Replay Dataset

Since SIMPL places an explicit prior on latent trajectories that are smooth and continuous I tested whether it could be used to model a dataset where the latent variable is non-continuous. For this I simulated a synthetic "replay" dataset from $N = 225$ small Gaussian place cells. In this dataset the latent variable and behaviour perfectly match except for regular, brief periods of "replay" where the latent variable jumps to a new location. Using the same hyperparameters as in the main text I found that SIMPL was able to recover the latent variable, capturing (or "decoding") the replay events with high accuracy

(fig. E.3), despite its smoothness prior.

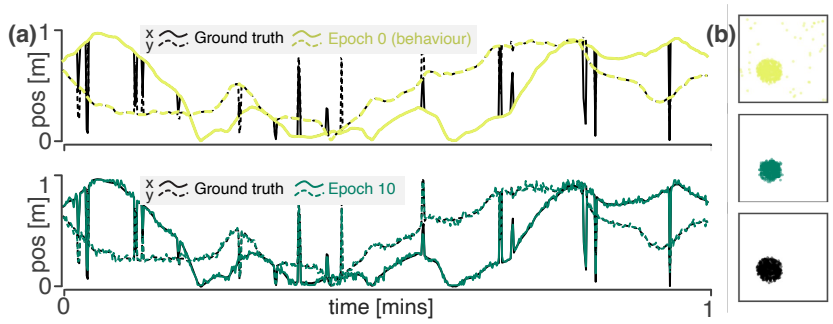

Figure E.3: A synthetic hippocampal "replay" dataset. **(a)** One minute of trajectory, x-coordinate in solid line, y-coordinate in dashed. The behaviour (light-green, top panel) is smooth, actually matching the latent most of the time except when the latent takes regular, brief discontinuous jumps reminiscent of hippocampal replay events. After optimisation SIMPL is able to recover the latent (dark-green, bottom panel) and capture the replay events with high accuracy. **(b)** Spike raster plots; spikes plotted against the behaviour, optimised latent and ground truth latent.

E.5.4 Automatic Place Field Detection

In Figure 5.3, it was shown that the tuning curves of place cells in the hippocampus undergo statistically significant changes when optimised using SIMPL. For this analysis, individual place fields were automatically identified from the binned rate maps as isolated regions of elevated activity within a cell's tuning curve. This was done by thresholding the activity of each neuron at 1 Hz and identifying contiguous regions of activity with a peak firing rate above 2 Hz and a total area less than half that of the full environment, similar to approaches taken in previous work (Tanni et al. 2022).

E.6 Summary Table of Related Methods

Some of the most relevant LVM and dimensionality reduction techniques are summarized here in the context of our five key desiderata as described in the related work section. These are:

1. Complex tuning curves: Does the model learn/infer non-linear tuning curves as opposed to linear/exponential-linear/etc. tuning curves.

2. Smooth latent dynamics: Does the model impose smooth temporal dynamics on the latent space (e.g. by assuming a linear dynamical system, Gaussian process or using an RNN), as opposed to treating each time point independently.

3. Spike-friendly: Was the method designed for spiking data. For probabilistic models, this refers to whether the generative noise model is Poisson as opposed to, say, Gaussian.

4. Exploit behaviour: Does/can the model use behaviour (as an observation, contrastive loss-target, initialisation, or otherwise) to guide latent discovery.

5. Scalable: Can the model scale to datasets of long duration. Specifically, in available open-source implementations of the method does training/inference have near-linear time complexity. Note this does *not* mean that compute time is necessarily fast in an absolute sense, just that scaling is linear.

Table E.3: A table of comparable models and their properties. N/A means the criterion is not applicable to the model. N/S means the criterion is not specified or may be dependent on implementation specifics. Techniques in **bold** are compared to on my benchmark dataset in fig. 5.6

Model	**Complex tuning curves**	**Smooth latent dynamics**	**Spike-friendly**	**Exploit behaviour**	**Scalable**
SIMPL (My method)	Yes	Yes	Yes	Yes	Yes
GPLVM (Lawrence 2003)	Yes	N/S	No	Yes[§]	Yes[#]
P-GPLVM (Wu et al. 2017)	Yes	Yes	Yes	Yes[§]	No
M-GPLVM (Jensen et al. 2020)	Yes	No	No	No	Yes[#]
faeLVM (Bjerke et al. 2023)	Yes[†]	Yes	Yes	No	Yes
PfLDS (Gao et al. 2016)	Yes	Yes	Yes	No	Yes

Continued on next page

Table E.3 – continued from previous page

Model	Complex tuning curves	Smooth latent dynamics	Spike-friendly	Exploit behaviour	Scalable
VIND (Hernandez et al. 2018)	Yes	Yes	Yes	No	Yes
pi-VAE (Zhou et al. 2020)	Yes	No	Yes	Yes	Yes
CEBRA (Schneider et al. 2023)	N/A	No	Yes	Yes	Yes
MIND (Low et al. 2018)	Yes	No	No	No	Yes
LFADS (Pandarinath et al. 2018)	No	Yes	Yes	Yes	Yes
TNDM (Hurwitz et al. 2021)	No	Yes	Yes	Yes	Yes
GP-SDEs (Duncker et al. 2019)	No	Yes	N/S	N/S	Yes
rSLDS (Linderman et al. 2016)	No	Yes	No	Yes§	Yes
gpSLDS (Hu et al. 2024)	No	Yes	No	Yes§	Yes
GPDM (Wang et al. 2005)	Yes	Yes	No	Yes§	No
MM-GPVAE (Gondur et al. 2023)	No	Yes	Yes	No	Yes
PSID (Sani et al. 2021)	No	Yes	No	Yes	Yes
GPFA (Yu et al. 2008a)	No	Yes	No	No	Yes#
P-GPFA (Nam 2015)	No	Yes	Yes	No	No
SSMDM (Zoltowski et al. 2020)	No	Yes	Yes	Yes	Yes
PLNDE (Kim et al. 2021)	No	Yes	Yes	Yes	Yes
GLDS (Kalman 1960)	No	Yes	No	No	Yes
DKF (Krishnan et al. 2015)	Yes	Yes	No	No	Yes
PLDS (Macke et al. 2011)	No	Yes	Yes	No	Yes
UMAP (McInnes et al. 2018)	N/A	No	No	No	No
TSNE (Maaten et al. 2008)	N/A	No	No	No	No
pPCA (Pearson 1901; Tipping et al. 1999)	No	No	No	No	Yes

Continued on next page

Table E.3 – continued from previous page

Model	Complex tuning curves	Smooth latent dynamics	Spike-friendly	Exploit behaviour	Scalable
dPCA (Kobak et al. 2016)	No	No	No	Yes	Yes

#: Scalable if using an implementation making induction point approximations.

§: Algorithm could be initialised at behaviour.

†: Assumes neurons in a given ensemble have a shared tuning curve structure (e.g. Gaussian) with neuron-specific transformations (e.g. shift and scale)

Appendix F

UCL Research Paper Declaration Forms

Paper 1

1. **For a research manuscript that has already been published** (if not yet published, please skip to section 2):

a) **What is the title of the manuscript?** *RatInABox, a toolkit for modelling locomotion and neuronal activity in continuous environments*

b) **Please include a link to or doi for the work:** `https://doi.org/10.7554/eLife.85274`

c) **Where was the work published?** *eLife*

d) **Who published the work?** *eLife Sciences Publications, Ltd.*

e) **When was the work published?** *February 9, 2024*

f) **List the manuscript's authors in the order they appear on the publication:** *Tom M. George, Mehul Rastogi, William de Cothi, Kimberly L. Stachenfeld, Claudia Clopath, Caswell Barry*

g) **Was the work peer reviewed?** *Yes*

h) **Have you retained the copyright?** *Yes, it was published under a Creative Commons Attribution 4.0 International License (CC BY 4.0)*

i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)?** *Yes, ht tp s:*

//www.biorxiv.org/content/10.1101/2022.08.10.503541

If 'No', please seek permission from the relevant publisher and check the box next to the below statement:

☐ I acknowledge permission of the publisher named under 1d to include in this thesis portions of the publication named as included in 1c.

2. **For a research manuscript prepared for publication but that has not yet been published** (if already published, please skip to section 3): *N/A*

3. **For multi-authored work, please give a statement of contribution covering all authors** (if single-author, please skip to section 4): *TMG conceived the project, designed the toolkit, implemented the code, and wrote the manuscript. MR contributed to code development. WDC, KS, CC, and CB advised.*

4. **In which chapter(s) of your thesis can this material be found?** *Chapter 1*

Paper 2

1. **For a research manuscript that has already been published** (if not yet published, please skip to section 2):

a) **What is the title of the manuscript?** *Theta Sequences as Eligibility Traces: A Biological Solution to Credit Assignment*

b) **Please include a link to or doi for the work:** `https://openreview.net/forum?id=vd16AYbem3Z`

c) **Where was the work published?** *ICLR 2023 TinyPapers*

d) **Who published the work?** *International Conference on Learning Representations (ICLR)*

e) **When was the work published?** *March 2023*

f) **List the manuscript's authors in the order they appear on the publication:** *Tom M. George*

g) **Was the work peer reviewed?** *Yes*

h) **Have you retained the copyright?** *Yes*

i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)?** *No, ICLR distributed but does not own the work. Ownership remains with me.*

If 'No', please seek permission from the relevant publisher and check the box next to the below statement:

☐ I acknowledge permission of the publisher named under 1d to include in this thesis portions of the publication named as included in 1c.

2. **For a research manuscript prepared for publication but that has not yet been published** (if already published, please skip to section 3): *N/A*

3. **For multi-authored work, please give a statement of contribution covering all authors** (if single-author, please skip to section 4): *Single-author.*

4. **In which chapter(s) of your thesis can this material be found?** *Chapter 2*

Paper 3

1. **For a research manuscript that has already been published** (if not yet published, please skip to section 2):

a) **What is the title of the manuscript?** *Rapid learning of predictive maps with STDP and theta phase precession*

b) **Please include a link to or doi for the work:** `https://doi.org/10.7554/eLife.80663`

c) **Where was the work published?** *eLife*

d) **Who published the work?** *eLife Sciences Publications, Ltd.*

e) **When was the work published?** *March 16, 2023*

f) **List the manuscript's authors in the order they appear on the publication:** *Tom M. George, William de Cothi, Kimberly L. Stachenfeld, Caswell Barry*

g) **Was the work peer reviewed?** *Yes*

h) **Have you retained the copyright?** *Yes, it was published under a Creative Commons Attribution 4.0 International License (CC BY 4.0)*

i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)?** *Yes,* `https://www.biorxiv.org/content/10.1101/2022.04.20.488882v2`
If 'No', please seek permission from the relevant publisher and check the box next to the below statement:

☐ I acknowledge permission of the publisher named under 1d to include in this thesis portions of the publication named as included in 1c.

2. **For a research manuscript prepared for publication but that has not yet been published** (if already published, please skip to section 3): *N/A*

3. **For multi-authored work, please give a statement of contribution covering all authors** (if single-author, please skip to section 4): *TMG and WDC conceived the project and designed the model. TMG implemented the code and performed the simulations. WDC and TMG wrote the manuscript. KS and CB advised on the project and manuscript. WDC and TMG contributed equally to this work.*

4. **In which chapter(s) of your thesis can this material be found?** *Chapter 3*

Paper 4

1. **For a research manuscript that has already been published** (if not yet published, please skip to section 2):

a) **What is the title of the manuscript?** *A generative model of the hippocampal formation trained with theta-driven local learning rules*

b) **Please include a link to or doi for the work:** `https://openreview.net/forum?id=yft4JlxsRf&noteId=Ozb895hT6L`

c) **Where was the work published?** *NeurIPS 2023*

d) **Who published the work?** *Conference on Neural Information Processing Systems (NeurIPS)*

e) **When was the work published?** *September 2023*

f) **List the manuscript's authors in the order they appear on the publication:** *Tom M. George, Kimberly L. Stachenfeld, Caswell Barry, Claudia Clopath, Tomoki Fukai*

g) **Was the work peer reviewed?** *Yes*

h) **Have you retained the copyright?** *Yes*

i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)?** *No, NeurIPS distributed but does not own the work. Ownership remains with me.*

If 'No', please seek permission from the relevant publisher and check the box next to the below statement:

☐ I acknowledge permission of the publisher named under 1d to include in this thesis portions of the publication named as included in 1c.

2. **For a research manuscript prepared for publication but that has not yet been published** (if already published, please skip to section 3): *N/A*

3. **For multi-authored work, please give a statement of contribution covering all authors** (if single-author, please skip to section 4): *TMG and TF conceived the project. TMG designed the model, implemented the code, and wrote the manuscript. KS, CB, CC, and TF advised.*

4. **In which chapter(s) of your thesis can this material be found?** *Chapter 4*

Paper 5

1. **For a research manuscript that has already been published** (if not yet published, please skip to section 2):

a) **What is the title of the manuscript?** *SIMPL: Scalable and hassle-free optimization of neural representations from behaviour*

b) **Please include a link to or doi for the work:** `https://openreview.net/forum?id=9kFaNwX6rv`

c) **Where was the work published?** *ICLR 2025*

d) **Who published the work?** *International Conference on Learning Representations (ICLR)*

e) **When was the work published?** *January 2025*

f) **List the manuscript's authors in the order they appear on the publication:** *Tom M. George, Pierre Glaser, Kimberly L. Stachenfeld, Caswell Barry, Claudia Clopath*

g) **Was the work peer reviewed?** *Yes*

h) **Have you retained the copyright?** *Yes*

i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)?** *Yes, https://www.biorxiv.org/content/10.1101/2024.11.11.623030v1*

If 'No', please seek permission from the relevant publisher and check the box next to the below statement:

☐ I acknowledge permission of the publisher named under 1d to include in this thesis portions of the publication named as included in 1c.

2. **For a research manuscript prepared for publication but that has not yet been published** (if already published, please skip to section 3): *N/A*

3. **For multi-authored work, please give a statement of contribution covering all authors** (if single-author, please skip to section 4): *TMG conceived the project, designed the method, and implemented the code. PG contributed to code development. TMG and PG wrote the manuscript. KS, CB, and CC advised.*

4. **In which chapter(s) of your thesis can this material be found?** *Chapter 5*

Signatures

e-Signatures confirming that the information above is accurate (this form should be co-signed by the supervisor/ senior author unless this is not appropriate, e.g. if the paper was a single-author work):

Candidate: Tom George  **Date:** *June 26, 2025*

Supervisor: Prof. Caswell Barry  **Date:** *June 26, 2025*