A Biologically Plausible, Dynamics-based Model of Memory Using Neurosymbolic Methods

by

Jakeb Chouinard

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2026

© Jakeb Chouinard 2026

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

The following chapters contain material adapted from papers on which I was the first author:

1. Chapters 1, 2, and 4-8 include content adapted from: Jakeb Chouinard and Chris Eliasmith. **A neural model of short-term and intermediate-term memory using neuro-symbolic representations for semantic and spatial memory tasks**. In *Proceedings of the Annual Meeting of the Cognitive Science Society 2026 (CogSci 2026)*, Brazil, July 2026. I was responsible for the ideation, implementation, and experimentation as well as for writing and revising this work. Co-author Chris Eliasmith provided conceptual and editorial suggestions as well as supervision throughout.

Abstract

This thesis presents a novel model of memory that addresses previous gaps in working memory modelling. Traditionally, models of memory treat memory as an atemporal process; they present a vector or trace of a memory that only changes with item presentations. We instead provide a dynamics-based model of short-term memory using vector projection, and we further characterize its capacity and forgetting mechanisms using temporal decay and neural saturation. We also provide an auto-associative memory that uses bundles of high-dimensional vectors to implicitly associate items to each other, enabling bundle reconstruction from single components. Lastly, we modify the parameters of the short-term memory to create a primacy-biased integrator of task-contextual embeddings.

We validate our model and demonstrate its ability to generalize across domains by implementing it in a spiking neural network and simulating semantic and spatial memory tasks with minimal parameter changes between simulations. For each simulation, we represent stimuli (e.g., list items or spatial landmarks) and task embeddings (e.g., temporal position in a list or spatial position within a room) as Spatial Semantic Pointers, and we compare our model's results to human experimental results. Across the serial recall, free recall, and positional recall experiments, our model consistently demonstrates human-like performance, demonstrating no statistically significant difference for 72.5% of compared data-points and negligible-to-small effect sizes for 91.2% of compared data points. Across all data points, we find mean and median absolute effect sizes of $|\bar{h}| \in (0.125, 0.203)$ and $|\hat{h}| \in (0.082, 0.189)$ respectively, suggesting significant coherence between model and experimental task results.

Acknowledgements

To start, I would like to acknowledge my supervisor, Dr. Chris Eliasmith, for his guidance and for the freedom I was granted to pursue my interests in my research. I am deeply grateful for our many conversations discussing the nitty-gritty details of my work, including the many tangents I explored. I would also like to thank those who I have had the pleasure of connecting with throughout my time as a member of the Centre for Theoretical Neuroscience (CTN) and the Computational Neuroscience Research Group (CNRG), namely Dr. Terrence Stewart, Dr. Michael Furlong and Dr. Britt Anderson. Our discussions have been central to my development as a researcher, allowing me to branch out further while creating meaningful connections within our field.

I would additionally like to thank my collaborators and colleagues within the CTN, the CNRG, and otherwise. It has truly been a pleasure to learn from your research, as well as to share my own research with you all. I will always be grateful for the questions asked and the answers provided.

Lastly, I would like to thank my loved ones and my friends—past and present. It is your support and kindness that has emboldened me to pursue my passions above all else. In me lies a piece of each of you, and for that, I am thankful.

J.C.

Dedication

This work is dedicated to my mother, who in face of the storm, taught me to stand firm.

Table of Contents

Author's Declaration ii

Statement of Contributions iii

Abstract iv

Acknowledgements v

Dedication vi

List of Figures x

List of Tables xii

1 Introduction 1

1.1 Memory 2

1.1.1 Multiscale Memory 4

1.2 Toward Dynamic Memory Systems 5

2 Background 9

2.1 Semantic Pointer Architecture 9

2.1.1 Holographic Reduced Representations 10

2.2 Structured Feature Representation 12

2.2.1 Spatial Semantic Pointers 13

2.3 Working Memory Tasks 15

2.3.1 Semantic Memory Tasks . . . 15
2.3.2 Spatial Memory Task . . . 16
2.4 Neural Realization . . . 17
2.4.1 Neural Architectural Diagrams . . . 18

3 Related Work **20**
3.1 Mathematical Models . . . 20
3.1.1 Ordinal Serial Encoding Model . . . 20
3.1.2 SSP-Attend . . . 21
3.1.3 Non-Associative Additive Model . . . 22
3.1.4 Temporal Context Model . . . 23
3.1.5 Context Maintenance and Retrieval Model . . . 25
3.2 Artificial Neural Models . . . 25
3.2.1 Tolman-Eichenbaum Machine . . . 25
3.2.2 Modern Hopfield Networks . . . 25
3.2.3 Entity Network . . . 26
3.3 Spiking Neural Models . . . 26
3.3.1 Spiking Neural Ordinal Serial Encoding Model . . . 26
3.3.2 Context Unified Encoding Model . . . 27
3.4 Summary . . . 27

4 Short-term Memory Modelling **29**
4.1 Temporal Dynamics . . . 29
4.2 Bounded Representations . . . 32
4.3 Capacity . . . 36
4.3.1 Remembering Chunks . . . 37
4.3.2 Neural Saturation Dynamics . . . 39

5 Intermediate-term Memory Modelling **44**
5.1 Implicit Auto-Associative Mappings . . . 44
5.2 Temporal Integration . . . 47

6 Semantic Memory Tasks **56**
6.1 Methodology 56
6.2 Serial List Recall 57
6.2.1 Results 60
6.3 Free List Recall 64
6.3.1 Results 64

7 Spatial Memory Task **70**
7.1 Methodology 70
7.1.1 Results 74

8 Results Summary and Conclusion **77**
8.1 Results Summary 77
8.2 Conclusion 78

References **81**

APPENDIX **86**

A Results Tables **87**

List of Figures

1.1 Example serial position curves in list recall experiments 3
1.2 A simplified diagram of the memory model 6

2.1 An overview of symbols used in architectural diagrams 19

3.1 A diagram showing the update processes of the Temporal Context Model . 24

4.1 A comparison of the OSE and snOSE to a recurrent dynamic system . . . 33
4.2 An analytical simulation of the projection-based memory system 37
4.3 A visualization of representational saturation in neural populations 39
4.4 An architectural diagram of the STM network 41
4.5 A comparison of the analytical satproj memory model and the projection memory model realized in spiking neurons 43

5.1 An architectural diagram of the ITM_L network 47
5.2 A demonstration of learning implicit associations with the AML in spiking neurons 48
5.3 An architectural diagram of the ITM_p network 51
5.4 A comparison of the analytical and spiking neural implementations for the ITM_p 52
5.5 A demonstration of the revised analytical ITM_p with an evident primacy bias 53
5.6 A comparison of spiking neural ITM_p behaviours for varying list lengths and gain parameters 55

6.1 An architectural diagram of the integrated models for semantic memory tasks 58
6.2 Model and experimental serial recall results 61

6.3 Confidence intervals and absolute Cohen's h for proportion differences between model and experimental serial recall results 62
6.4 A comparison of STM+ITM, snOSE, and CUE models to experimental serial recall results 63
6.5 Model and experimental free recall results 66
6.6 Confidence intervals for proportion differences between model and experimental free recall results 67
6.7 Absolute Cohen's h for proportion differences between model and experimental free recall results 67
6.8 A comparison of STM+ITM and CUE models to experimental free recall results 69

7.1 An architectural diagram of the integrated models for spatial memory tasks 72
7.2 Model and experimental spatial recall results 75
7.3 A comparison of STM+ITM SLAM and snSLAM models to experimental spatial recall results 76

8.1 Distribution of absolute effect sizes for results across tasks 78

List of Tables

6.1 Model Parameters for Semantic Memory Tasks 59

7.1 Model Parameters for the Spatial Memory Task 73

A.1 Summary of Data Analyses for Serial Recall Results 87
A.2 Summary of Data Analyses for Free Recall Results 88
A.3 Summary of Data Analyses for Spatial Recall Results 89

Chapter 1

Introduction

> *If we wish to know about a man, we ask "what is his story—his real, inmost story?"—for each of us is a biography, a story. Each of us is a singular narrative, which is constructed, continually, unconsciously, by, through, and in us—through our perceptions, our feelings, our thoughts, our actions; and, not least, our discourse, our spoken narrations. Biologically, physiologically, we are not so different from each other; historically, as narratives, we are each of us unique.*
>
> Oliver Sacks, *The Man Who Mistook His Wife For A Hat* (1985)

Memory is a vital part of cognition. Its many parts determine how one learns, plans, and ultimately experiences the world around them. As a bridge between short-term memory and long-term memory, intermediate-term memory allows for semi-stable memories to be rapidly encoded through short-term synaptic plasticity. Throughout this thesis, we pursue a dynamics-based understanding of short-term and intermediate-term memory systems to develop a general model of working memory. Working memory operates on the timescales of both short-term and intermediate-term memory, suggesting a complex interaction of activity-based and weight-based representations that cooperate to store and recall information.

Previous efforts in modelling memory treat the formation of a memory trace as an atemporal process—the memory is modified only when items are freshly introduced. An atemporal memory trace is inconsistent with human memory, particularly when we consider how it must be realizable in the biological substrate of the brain. This thesis focuses on two core questions: How can we formulate a temporally dynamic model of human-like working memory? And, how can we verify this model using biologically-plausible neural

simulation? Building on prior work, we use Holographic Reduced Representations—a form of Vector Symbolic Algebra—to implement sub-symbolic representations with symbol-like operations. We further implement our model in spiking neurons and compare its results to human behavioural results in several different memory tasks.

This chapter situates our efforts in relation to neurobiological cognition. Cognition is highly dependent on learning and memory as general phenomena; however, both learning and memory are expressed on multiple different time-scales in multiple different brain areas. Here, we focus on some of the key learning and memory processes driving working memory. In doing so, we identify a path forward in considering these biological systems as dynamic systems.

1.1 Memory

Discussion and characterization of memory has long been a topic of discussion amongst natural philosophers and scientists. As early as the Ancient Greeks, Aristotle recognized and documented the importance of list structure and order in remembering and recalling lists (Aristotle, 350 B.C.E./1930). Similarly, Cicero—an Ancient Roman rhetorician—championed the importance of the introduction and conclusion of speeches, claiming that they are the most salient and memorable portions (Cicero, 55 B.C.E./1942). Both the claims put forward by Aristotle and Cicero find themselves supported by a wide array of modern working memory experiments, demonstrating clear patterns of behaviour across modalities and experimental paradigms.

First formally examined by Ebbinghaus (1885/1913), recollection of a sequence of syllables exhibited key characteristics across lists, such as serial position effects, list repetition effects, and coherence effects. Serial position effects refer to how the position of an item in a sequence affects its likelihood of being recalled. Particularly, Ebbinghaus found that earliest and latest items within the sequence were most likely to be recalled. While Ebbinghaus only tested on himself, early experimental psychology studies, such as those performed by Deese and Kaufman (1957) and Murdock (1962), replicated these results across groups of participants. Across these studies, participants recalling sequences of words demonstrated the typical $\cup$-shaped probability of recollection Ebbinghaus identified, where earliest and latest items are best remembered. The tails of the $\cup$-like probability have since become known as primacy and recency effects. Primacy and recency effects reflect the average individual's increased ability to recall the earliest and latest pieces of information in a sequence, respectively. Additional studies also supported Ebbinghaus' discovery of list repetition effects—the increase in ability to recall items of a list when the item is repeated

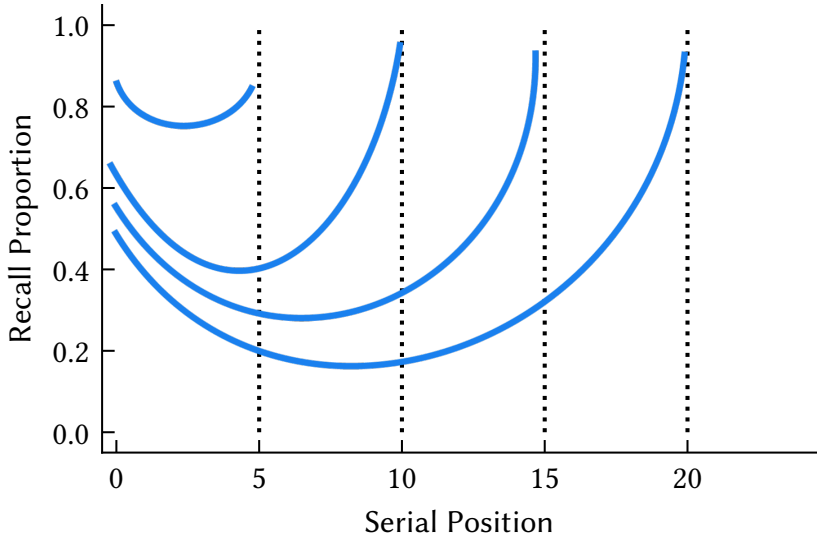


Figure 1.1: *Example serial position curves in list recall experiments.* These curves show simplified behaviours based on the curves reported in Murdock (1962). The distinctive ∪-shaped curves highlight the increased likelihood of recalling early and late list items in list recall experiments. Note: these curves are based on experimental data but do not reflect any particular experiment or published finding; they are intended to illustrate common trends in human list recall.

between or within lists (Hebb, 1961)—and coherence effects—the increased likelihood to recall conceptually coherent list items regardless of list position (Jenkins & Russell, 1952). Figure 1.1 shows a diagram of serial position curves typical of humans completing list recall tasks for varying list lengths. These curves are based on those reported in Murdock (1962) and highlight primacy and recency effects: a higher proportion of recalled list items between trials are at either the beginning or end of the to-be-recalled lists.

Key characteristics of sequential memory persist outside of semantic tasks as well. Spatial navigation is a critical part of cognition, as knowledge of the environment is essential for planning and executing tasks. Several studies demonstrate that in healthy adults, there are primacy and recency effects in recalling spatial position of landmarks observed during a navigation task (Hilton et al., 2023; Laczó et al., 2021). Similar to semantic tasks, the magnitudes of the primacy and recency effects depend on the experimental paradigm, such as a delay or a distractor task between the last item and recall. For a more comprehensive review of experiments in psychology that highlight serial position effects, list repetition effects, and coherence effects, see Kahana (2017).

1.1.1 Multiscale Memory

Memory is typically divided into two distinct classes: short-term and long-term memory. Short-term memory refers to highly transient sensory memory as well as attention-dependent working memory. By contrast, long-term memory refers to stable memories in general. Stable memories take the form of memories of lived experiences, i.e., episodic memory, and memories of pieces of information, i.e., semantic memory. Long-term memory can also be used to refer to procedural or habitual memory, where one can perform an action without conscious thought due to their expertise, such as finding individual keys on a keyboard when typing. Within humans, sensory memory and working memory implicate corresponding sensory cortices and the prefrontal cortex, respectively. While sensory and working memory are treated as more localized processes, long-term memory implicates a variety of medial temporal lobe and cortical structures (Camina & Güell, 2017).

The timescale of each form of memory is strongly related to the neurobiological mechanisms by which they form. Short-term memory is most commonly thought to be encoded in neural populations' activities, also referred to as "engrams" in relevant brain areas (Poo et al., 2016). Sensory memory is the creation of some engram that represents or encodes a stimulus; however, once that stimulus is removed, the representation in the corresponding sensory cortex rapidly decays. Working memory decays similarly to sensory memory, although on a significantly slower timescale. Attentional mechanisms can also be used to extend working memory representations through overt or covert rehearsal mechanisms (e.g., repeating portions of a phone number out loud or in one's head when trying to remember it; Kamiński et al., 2017). Long-term memory operates on a significantly slower timescale than short-term memory, however, requiring consolidation processes to embed stable memories into synaptic connections via long-term synaptic plasticity (LTSP; Goto, 2022). We refer to LTSP as an umbrella term for long-term potentiation and long-term depression, where synapse connections are strengthened or weakened depending on neuronal activities.

While this account of memory provides a convenient distinction between short and long-term memory, it fails to provide an account of the in-between. That is to say, how might memory representations persist over the course of an hour without overt rehearsal, covert rehearsal, or consolidative processes? Rapid synapse-based learning in the hippocampus provides an account of intermediate-term memory where short-term synaptic plasticity (STSP) rapidly embeds experiences into hippocampal neural connections (Kamiński, 2017). Similar to LTSP, we refer to STSP for both short-term potentiation and short-term depression, both of which rapidly change neural connectivity. By this account, the brain can pool sensory and working memory representations into a coherent, semi-stable form that is capable of being consolidated once an individual goes to sleep or enters a restful state

(Wamsley, 2019; Wilson & McNaughton, 1994). The notion of intermediate-term memory obfuscates that of working memory slightly. Instead of a persistent activity state, working memory with STSP broadens its timescale and bridges short-term memory to long-term memory. In doing so, working memory would depend on both short-term, activity-based memory and intermediate-term, STSP-based memories.

1.2 Toward Dynamic Memory Systems

When viewing working memory as a set of interacting systems relying on multiple different forms of representation and timescales, an important question arises: how can we understand and characterize the dynamics of these interacting systems? Further, how can we do so within a neurally-plausible framework? This thesis formulates a dynamic model of working memory incorporating both short-term and intermediate-term memory representations. We place particular emphasis on generalizability, as it is important for any coherent model of memory that its proposed techniques can be used across modalities—i.e., the same memory model capable of performing a list recall task should be capable of performing spatial memory tasks.

While previous modelling efforts have worked towards this kind of generalizability, few models have prioritized neural plausibility (See Sections 3.1 and 3.2). Past models commonly use forms of backpropagation and rate neurons rather than Hebbian learning processes and spiking neural models. In contrast, our model is implemented entirely in spiking neurons, and all learning is done through Hebbian learning processes, making it uniquely neurally-plausible.

We introduce our model here in brief to help facilitate understanding throughout the remainder of this thesis. Our model is composed of three parts based on short-term and intermediate-term memory. Figure 1.2 shows a simplified architectural diagram of our complete model. We use Holographic Reduced Representations (see Chapter 2) to represent to-be-remembered items, $\boldsymbol{\phi}_1$, as well as their positions (temporal or spatial), $\boldsymbol{\phi}_2$, in the working memory tasks. Items are presented and associated with their positions through binding, $\circledast$, and/or bundling, Σ, depending on the downstream memory component. Once presentation is complete, we query the memory components using $\boldsymbol{\phi}_q$, which can be any value of $\boldsymbol{\phi}_1$ or $\boldsymbol{\phi}_2$ depending on what we want to recall. Our model implements two activity-based memory stores: the short-term memory model (STM) maintains a recency-biased, activity-based memory, $\boldsymbol{m}$, and the temporal integrator of the intermediate-term memory (ITM_p) integrates positional information to maintain a primacy-biased, activity-based task context, $\boldsymbol{p}$. Our model also implements one weight-based memory: the association matrix

of the intermediate-term memory (ITM_L) performs one-shot, Hebbian learning to associate items and positions as they are presented/observed throughout each task. Whereas the STM can be queried directly using positions or items, the ITM_p is used in addition to $\boldsymbol{\phi}_q$ to query the ITM_L during recall, as the associations stored within the weight-based memory cannot be accessed without some input vector.

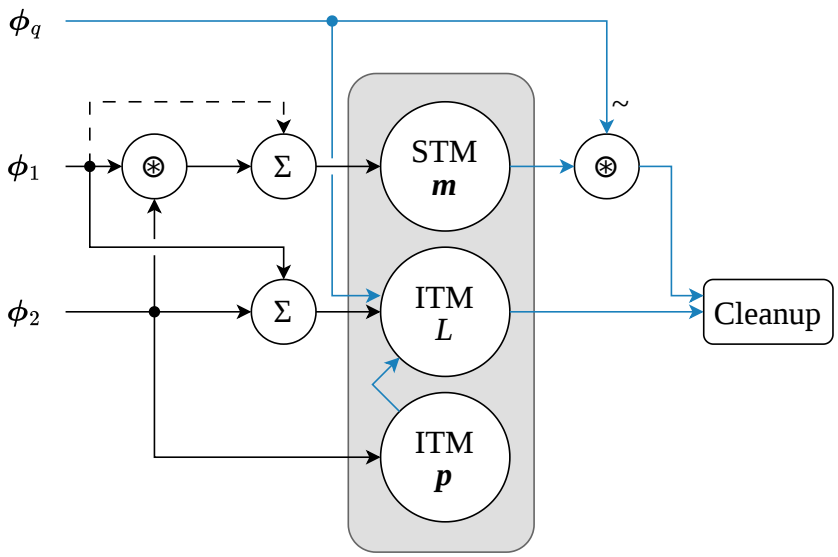


Figure 1.2: *A simplified diagram of the memory model.* Our model contains three parts: the short-term memory (STM), the positional integrator of the intermediate-term memory (ITM_p), and the association matrix of the intermediate-term memory (ITM_L). ϕ_1 denotes input features vectors, such as list items; ϕ_2 denotes "positional" vectors, such as when or where something was observed in a task; and ϕ_q denotes query vectors used for prompting the model during recall. Black lines show presentation-time model interactions with dashed lines being specific to serial and free recall tasks. Blue lines show recall-time model interactions. Reproduced from *A neural model of short-term and intermediate-term memory using neuro-symbolic representations for semantic and spatial memory tasks* (Chouinard & Eliasmith, 2026). Copyright retained by the authors.

We demonstrate our model's efficacy by comparing to human experimental results across three different working memory tasks. Specifically, we have the model perform a serial recall task in which participants must recall 10 items in order of their appearance without delay (Jahnke, 1968); a free recall task in which participants must recall 12 words

without delay while orating the most recently presented word to prevent rehearsal (Howard & Kahana, 1999); and a spatial recall task in which participants must recall the positions of 2-4 items encountered in an augmented reality (AR) environment following a chase-based distractor task (Maidenbaum et al., 2025).

With just under 1.5 million neurons, our model can replicate human-like patterns in all recall tasks, including serial position effects, positional accuracy, and first and conditional recall probabilities. Further, our model leverages novel projection-based dynamics and is the first examination, to our knowledge, of representational capacity due to finite neural expressivity in a Semantic Pointer Architecture-based spiking neural model. Our model also leverages continuous feature embeddings in a novel way to realize human-like biases in working memory tasks, particularly as it concerns the likelihood of an individual to recall a list item nearer to previously recalled list items. Lastly, our model is constructed with a generalizable architecture, allowing us to use it for list recall as well as map learning without changing the underlying architecture. Overall, our novel dynamic system provides a neurally-plausible account of short-term and intermediate-term memory, paving the way for generalizable and scalable ways to implement memory in both analytical and spiking neural models.

The remainder of this thesis is structured as follows:

- Chapter 2 provides necessary background on: the neurally-plausible symbol-like representations used throughout this paper; the representational structure of stimuli for semantic and spatial recall tasks; and the transition from analytical dynamic systems to spiking neural populations.

- Chapter 3 gives a comprehensive overview of related work, including prior analytical and neural models, to situate our model in the literature.

- Chapter 4 focuses on the development of the short-term memory model (STM), highlighting temporal dynamics and saturation-based capacity limitations.

- Chapter 5 explains the functionality of the intermediate-term memory model (ITM_L and ITM_p), focusing on the processes of learning implicit associations and integrating positional information.

- Chapter 6 describes the architecture and results for the semantic recall tasks, illustrating how our short-term and intermediate-term memory models interact with external control signals and other task-related networks to perform list recall as well as comparing to previous spiking neural models.

- Chapter 7 describes the architecture and results for the spatial recall task, illustrating how the short-term and intermediate-term memory models were embedded into the snSLAM model as well as comparing to the original snSLAM model.
- Chapter 8 pools the results from Chapters 6 and 7 to evaluate the performance of the model; we highlight the model's ability to exhibit human-like behaviour, summarize the contributions of this work, and discuss potential avenues of future research.

Chapter 2

Background

The focus of this thesis is on a dynamic model for short-term and intermediate-term memory using distributed representations. We use the Semantic Pointer Architecture and Spatial Semantic Pointers as a representational basis for information in our model, leveraging their semi-symbolic nature to create and use associations between vectors. Within this chapter, we introduce the Semantic Pointer Architecture, focusing on the theoretical basis of Holographic Reduced Representations, including their extension to continuous spatial representations. We additionally highlight how information is represented within the semantic and spatial memory tasks to which our model will be subjected.

2.1 Semantic Pointer Architecture

The Semantic Pointer Architecture (SPA) proposes the use of high-dimensional vectors carrying semantic meaning, or Semantic Pointers (SPs), to construct models of cognition (Eliasmith, 2013). Individual cognitive SPs are unit-length and real-valued such that they can be easily transformed and manipulated in relation to a given model. Under the SPA, we can represent some set of concept instances, $\mathcal{C}$, by generating mappings between concept labels and connectionist representations. The set of resultant SPs can be more formally written as a set of concept-to-vector mappings:

$$\Psi = \left\{ \boldsymbol{\psi}(\cdot) : \mathcal{C} \rightarrow \mathbb{R}^d \right\} \tag{2.1}$$

where $\boldsymbol{\psi}(c) \in \mathbb{R}^d$ would be the high-dimensional embedding (SP) of concept instance c.

Most commonly, cognitive SPs are generated for some c by randomly sampling a point on the surface of the unit hypersphere. When sampled in a high-dimensional space, it is

highly unlikely for any two randomly sampled points to be significantly "similar"; however in particularly unlucky cases, it is possible to accidentally generate highly-similar—or even opposite—vectors. To prevent this, we specify some maximum similarity threshold, ϵ, such that for any two distinct mappings in Ψ, the absolute value of the inner-product—or the dot-product similarity—is below ϵ. For a generic set of SPs as defined in Equation 2.1, we write this as:

$$\Psi = \left\{ \boldsymbol{\psi}(\cdot) : \mathcal{C} \to \mathbb{R}^d \;\middle|\; \|\boldsymbol{\psi}(c_1)\|_2 = 1 \text{ and } \left|\boldsymbol{\psi}(c_1)^\top \boldsymbol{\psi}(c_2)\right| < \epsilon \;\forall\; c_1, c_2 \in \mathcal{C}, c_1 \neq c_2 \right\}$$

where $\|\cdot\|_2$ is the ℓ-2 norm, $(\cdot)^\top$ is the vector transpose, and $|\cdot|$ is the absolute value.

2.1.1 Holographic Reduced Representations

Holographic Reduced Representations (HRRs; Plate, 1995) have commonly been used to implement the SPA, allowing for the formation of structured, compositional representations of SPs. HRRs are a Vector Symbolic Algebra (VSA) that compress outer tensor products for creating dimensionality-preserving association structures based on the work done by Smolensky (1990). In doing so, HRRs ensure that nested compositional representations do not explode in dimensionality—a natural consequence of repeated outer tensor product operations. VSAs typically have five distinct operations that enable forming and manipulating compositional representations: *similarity* $(\cdot)$ is used to evaluate the semantic similarity between two representations using their geometric similarity; *bundling* $(+)$ is used to combine vectors such that the resultant vector is similar to its constituent vectors; *binding* $(\circledast)$ is used to combine vectors such that the resultant vector is dissimilar to its constituent vectors; *braiding* $(\bowtie)$ is used to provide ordering to structure representations; and *inversion* $(\cdot^\sim)$ is used to recover a vector from a bound set of other vectors.

The HRR VSA used by the SPA includes all of these operations except braiding—which is addressed in Section 2.2. In the HRR, similarity is implemented as vector inner products—or dot products; bundling is implemented as vector addition; binding is implemented as circular convolution; and inversion is implemented as the reversal of a vector's 2^{nd} to d^{th} entries. Inversion in this case allows for an approximate undoing of the circular convolution operation, which is a form of compressed tensor product. For two SPs,

$\boldsymbol{\psi}_1, \boldsymbol{\psi}_2 \in \mathbb{R}^d$, the HRR operations can be written as:

$$\begin{aligned}
\boldsymbol{\psi}_1 \cdot \boldsymbol{\psi}_2 &= \sum_{i=1}^{d} \left[\boldsymbol{\psi}_1 \odot \boldsymbol{\psi}_2\right]_i \\
\boldsymbol{\psi}_1 + \boldsymbol{\psi}_2 &= \boldsymbol{\psi}_1 + \boldsymbol{\psi}_2 \\
\left[\boldsymbol{\psi}_1 \circledast \boldsymbol{\psi}_2\right]_i &= \sum_{k=1}^{d} \left[\boldsymbol{\psi}_1\right]_k \left[\boldsymbol{\psi}_2\right]_{\mathrm{mod(i-k,n)+1}} \\
\boldsymbol{\psi}_1^{\sim} &= \left[\left[\boldsymbol{\psi}_1\right]_1, \left[\boldsymbol{\psi}_1\right]_d, \left[\boldsymbol{\psi}_1\right]_{d-1}, \ldots, \left[\boldsymbol{\psi}_1\right]_2\right]^\top
\end{aligned}$$

where $\left[\boldsymbol{\psi}\right]_i$ is the value of the i^{th} entry of $\boldsymbol{\psi}$; $\odot$ is the Hadamard, or element-wise, product; and mod is the modulo operation such that $\mathrm{mod}(x, n) = x - n\lfloor x/n \rfloor$ given $\lfloor \cdot \rfloor$ is the floor function. Note that the definition of the similarity measure is equivalent to the inner product, or the dot product, between $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_2$: $\boldsymbol{\psi}_1^\top \boldsymbol{\psi}_2$. We use these notations interchangeably. By examining the definitions for binding and inverting, we make note of the HRR identity vector, $\boldsymbol{\psi}_{\mathbb{I}} = [1, 0, 0, \ldots, 0]^\top$. The identity vector has one key property: it does not modify a vector when bound with it. By consequence, $\boldsymbol{\psi}_1 \circledast \boldsymbol{\psi}_{\mathbb{I}} = \boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_1 \circledast \boldsymbol{\psi}_1^{\sim} \approx \boldsymbol{\psi}_{\mathbb{I}}$.

Compositional structures using HRRs—"frames" per Plate (1995)—are typically formed with slot-filler representations. The slot-filler structure allows for multiple orthogonal vectors to be associated with a single relevant vector for consistency between frames. One common example of a slot and its fillers is $\boldsymbol{\psi}$(COLOUR) and the related embeddings of different colours (e.g., $\boldsymbol{\psi}$(RED)). As an example of how HRRs can be used to represent compositional structures, we consider the sentence: "The blue dog is to the left of the red house":

$$\begin{aligned}
\boldsymbol{\psi}(S) = &\left(\boldsymbol{\psi}(\text{BLUE}) \circledast \boldsymbol{\psi}(\text{COLOUR}) + \boldsymbol{\psi}(\text{LEFT}) \circledast \boldsymbol{\psi}(\text{POS}'\text{N})\right) \circledast \boldsymbol{\psi}(\text{DOG}) + \\
&\left(\boldsymbol{\psi}(\text{RED}) \circledast \boldsymbol{\psi}(\text{COLOUR}) + \boldsymbol{\psi}(\text{RIGHT}) \circledast \boldsymbol{\psi}(\text{POS}'\text{N})\right) \circledast \boldsymbol{\psi}(\text{HOUSE})
\end{aligned} \tag{2.2}$$

We can consider the frame $\boldsymbol{\psi}(S)$ as a composite of two frames—the frame containing slot-filler representations relevant to DOG and the frame containing slot-filler representations relevant to HOUSE. To access specific attributes of specific frames, we can query $\boldsymbol{\psi}(S)$ from Equation 2.2 using the relevant inverted vectors. For instance, we can query for the colour of the dog as follows:

$$\boldsymbol{\psi}(\text{Q}) = \boldsymbol{\psi}(S) \circledast \boldsymbol{\psi}^{\sim}(\text{DOG}) \circledast \boldsymbol{\psi}^{\sim}(\text{COLOUR})$$

where $\boldsymbol{\psi}$(Q) is the vector result of our query. The similarity between $\boldsymbol{\psi}$(Q) and $\boldsymbol{\psi}$(BLUE) will be approximately 1, while the similarity between $\boldsymbol{\psi}$(Q) and every other vector will

be approximately 0. Since circular convolution is a compression of the tensor product, the binding operation is inherently lossy. While the nature and magnitude of the loss depends on the structure of the representation and the dimensionality of the constituent representations themselves, it is likely that the similarity between a query's resultant vector and non-target vectors will be greater than ϵ. For this reason, queries are often paired with clean-up processes, such as an argmax over Ψ or lateral inhibition between neural networks representing individual vectors. These clean-up processes allow for only the most similar vector to be reported as the result of a query.

2.2 Structured Feature Representation

HRRs can be used to embed structured feature representations (i.e., structured slots) using self-binding. By binding a vector with itself, we create a vector with a structural relationship to the original vector. Repeated self-binding can be written as:

$$\boldsymbol{\psi}(\mathrm{N}) = \underbrace{\boldsymbol{\psi}(1) \circledast \boldsymbol{\psi}(1) \circledast \ldots \circledast \boldsymbol{\psi}(1)}_{\text{N times}}$$

which can be rewritten as $\boldsymbol{\psi}_1(N)$ where $\boldsymbol{\psi}_1$ denotes $\boldsymbol{\psi}(1)$ as the axis vector determining the self-binding structure. In this way, we can specify an arbitrary "axis" vector, and we can construct a lattice of evenly spaced points over some space. Consider a two-dimensional plane with the axis vectors $\boldsymbol{\psi}(X)$ and $\boldsymbol{\psi}(Y)$:

$$\boldsymbol{\psi}(x, y) = \boldsymbol{\psi}_X(x) \circledast \boldsymbol{\psi}_Y(y)$$

where $x, y \in \mathbb{Z}$. In this way, self-binding can be understood as a form of braiding in a VSA. In fact, if SPs are structured in such a way that they repeat after some N self-binding operations (i.e., $\boldsymbol{\psi}_\circ(N) = \boldsymbol{\psi}(\circ)$), they exhibit the same index-shift behaviour as is used for braiding in other VSAs. Unfortunately, random sampling of $\boldsymbol{\psi}(\cdot)$ from the surface of the unit n-ball does not guarantee the bound product of two random unit vectors will be unit length. In fact, a bound product may be longer or shorter than unit length depending on its Fourier coefficients. Because the length of a bound SP is not guaranteed, we might expect to see a vector have an excessively large or small magnitude if it is the result of several binding operations. That is to say, there are three possible outcomes for a large number of self-binding operations: $\lim_{N\to\infty} \|\boldsymbol{\psi}_X(N)\|_2 \in \{0, 1, \infty\}$.

To prevent unexpected magnitude changes due to binding, Plate (1995) examines binding in the Fourier domain: $\boldsymbol{\psi}_1 \circledast \boldsymbol{\psi}_2 = \mathcal{F}^{-1}\{\mathcal{F}\{\boldsymbol{\psi}_1\} \odot \mathcal{F}\{\boldsymbol{\psi}_2\}\}$. Since circular convolution

is equivalently an element-wise product in the Fourier domain, the behaviour of the magnitude for a large number of self-binding operations depends on the coefficients of the Fourier terms. By restricting the set of possible SPs to the set of unitary SPs—the set of SPs with Fourier coefficients of 1—binding results in a unit length vector. Using this subset of SPs, we can embed uniformly-divided spaces and create a semblance of coherent spatial representation; however we have so far restricted ourselves to integer binding.

2.2.1 Spatial Semantic Pointers

Spatial Semantic Pointers (SSPs) are an extension to the SPA based around the unitary subset of SPs. Since a unitary SP and its N^{th} self-binding can be rewritten as $\boldsymbol{\psi}_1 = \mathcal{F}^{-1}\left\{e^{j\boldsymbol{\theta}_1}\right\}$ and $\boldsymbol{\psi}_1(N) = \mathcal{F}^{-1}\left\{e^{jN\boldsymbol{\theta}_1}\right\}$ respectively where $\boldsymbol{\theta}_1 \in \mathbb{R}^d$ is a conjugate-symmetric phase vector of $\boldsymbol{\psi}_1$ and j is the imaginary unit, SSPs embed a continuous feature space (i.e., continuous slots) by allowing N to be any real number (Komer et al., 2019). For those familiar with Random Fourier Features (RFFs; Rahimi and Recht, 2007), SSPs and real-valued RFF embeddings are structurally identical—they both use distributions of embedding phases to project feature spaces into higher-dimensional embedding spaces. For some n-dimensional feature space, a d-dimensional SSP embedding is defined as:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \mathcal{F}^{-1}\left\{e^{j\Theta\boldsymbol{x}}\right\}$$

where $\Theta \in \mathbb{R}^{d\times n}$ is the conjugate-symmetric phase matrix for n feature spaces and $\boldsymbol{x} \in \mathbb{R}^n$ is the vector of the corresponding features. Since circular convolution in the Fourier space is the Hadamard product, SSP binding is equivalent to phase superposition for SSPs with different phase matrices and feature space superposition for SSPs with equivalent phase matrices:

$$\begin{aligned}\boldsymbol{\phi}_1(\boldsymbol{x}_1) \circledast \boldsymbol{\phi}_2(\boldsymbol{x}_2) &= \mathcal{F}^{-1}\left\{e^{j(\Theta_1\boldsymbol{x}_1+\Theta_2\boldsymbol{x}_2)}\right\}\\ \boldsymbol{\phi}(\boldsymbol{x}_1) \circledast \boldsymbol{\phi}(\boldsymbol{x}_2) &= \boldsymbol{\phi}(\boldsymbol{x}_1+\boldsymbol{x}_2)\end{aligned}$$

Because of their structure, SSPs induce a kernel over the embedded space. That is to say, any two points in the feature space, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, will have a dot product similarity of $K(\boldsymbol{x}_1 - \boldsymbol{x}_2)$ in the embedded space. Since SSPs typically represent each feature space dimension independently, the multidimensional kernel is a product of the kernels over each of the individual feature (Voelker, 2020):

$$K(\Delta\boldsymbol{x}) = \prod_{i=1}^{n} k_i\left([\Delta\boldsymbol{x}]_i\right)$$

and by the theorems laid out in Bochner (1959):

$$k_i\left([\Delta \boldsymbol{x}]_i\right) = \mathcal{F}^{-1}\left\{[\Theta \Delta \boldsymbol{x}]_i\right\} \approx \mathcal{F}^{-1}\left\{p([\boldsymbol{\omega}]_i\right\}$$

where $p([\boldsymbol{\omega}]_i)$ is the distribution of the sampled feature space's embedding phases for the i^{th} column of Θ. Since the phases of Θ are a finite sampling of $p(\boldsymbol{\omega})$, the induced kernel is an approximation of the underlying distribution.

Aside from representing independent, continuous spaces, we can construct sets of uniformly distributed SPs using the same embedding technique as SSPs. As previously mentioned, SPs are typically constructed by uniformly sampling points on the surface of the unit n-ball. Unitary SPs can be generated by taking these random points, converting them to Fourier representations, setting all coefficients to 1, and then converting them back to SPs; however, we can more efficiently construct a set of uniformly-distributed unitary SPs by constructing SSP embeddings and using each axis vector as an SP. As an example, consider generating two uniformly-distributed unitary SPs—$\boldsymbol{\psi}(a), \boldsymbol{\psi}(b) \in \mathbb{R}^d$. To generate these SPs, we create an SSP embedding for a 2-dimensional feature space. We then calculate $\boldsymbol{\psi}(a) = \mathcal{F}^{-1}\left\{e^{j\Theta[1,0]^\top}\right\}$ and $\boldsymbol{\psi}(b) = \mathcal{F}^{-1}\left\{e^{j\Theta[0,1]^\top}\right\}$, where the columns of Θ are independently sampled such that $\left(\boldsymbol{\psi}(a)\right)^\top \boldsymbol{\psi}(b) \approx 0$. For uniformly-distributed columns of Θ, we realize a uniform distribution of unitary SPs over the surface of the hypertorus, just as SPs are uniformly distributed over the surface of the hypersphere. By sampling uniformly-distributed unitary SPs from independent SSP embeddings, we find a more computationally efficient way to create sets of uniformly-distributed unitary SPs that are more compatible with SSPs during binding—i.e., the bound product will not change in magnitude.

Hexagonal Spatial Semantic Pointers

Hexagonal Spatial Semantic Pointers (HexSSPs; Dumont and Eliasmith, 2020) are a variation of SSPs that use grid cell-like frequency structure in their embeddings. By constructing Θ according to a simplex stacking-based paradigm, HexSSPs form a more efficient representation of multidimensional spaces. Each row of the Θ matrix is a differently oriented and scaled simplex in the n-dimensional feature space, resulting in embeddings of dimensionality $2N_R N_S(n+1)+1$ where N_R and N_S are the number of rotations and scales in the embedding. As N_R becomes sufficiently large, phase vectors—rows of Θ—approximate an equal distribution in every direction of the feature space. As N_S becomes sufficiently large, Θ approximates some radial distribution of scales and ultimately a symmetric probability density function in n dimensions. Because of their directional symmetry, HexSSPs realize a radial basis kernel where the embedding space similarity between two points is based

on the Euclidean distance between them. Most commonly, the HexSSPs in existing literature using a uniform radial distribution, inducing an integrated hypergeometric kernel (Chouinard, 2026). For sufficiently large N_R and N_S, the approximated HexSSP kernel for an n-dimensional feature space can be written as a power series of the form:

$$K(\Delta \boldsymbol{x}) \approx \sum_{k=0}^{\infty} \frac{(-1)^k \lambda^{-2k} \|\Delta \boldsymbol{x}\|_2^{2k}}{4^k (2k+1) \left(\frac{n}{2}\right)_k k!}$$

where λ is the length-scale parameter of the HexSSP embedding and $(\cdot)_k$ is the Pochhammer symbol such that $(x)_k = \prod_{i=0}^{k-1}(x+i)$.

2.3 Working Memory Tasks

Working memory is commonly tested through information recall experiments. In most experiments, participants are presented some set of items and are asked to recall them subject to some experimental conditions. The presentation of items can be from a list of words presented on a screen or from items that a participant sees in an environment. Typically, participants are asked to recall the items either in the order they were originally shown (serial recall) or in the order of their choice (free recall); however, they can also be asked for information regarding a random item's task-relevant information, such as its position in a list or its physical location in an environment. We compare our model's performance to human performance on three different working memory tasks in order to demonstrate the generality of the model's underlying principles.

2.3.1 Semantic Memory Tasks

In the serial and free recall memory tasks, participants are presented a list of items. To represent each of the items within a list of length N, we generate N unitary SPs as well as an SSP temporal embedding to capture the serial order of each list item. For each task, let Φ_V be the set of N orthogonal, unitary SPs representing each item in the list. Additionally, let Φ_T be the set of SSP embeddings corresponding to each item's serial order. We can write both of these as:

$$\Phi_V = \left\{ \boldsymbol{\phi}_V[i] \,|\, \boldsymbol{\phi}_V^\top[i] \boldsymbol{\phi}_V[k] \approx 0 \,\forall\, i \neq k,\, \boldsymbol{\phi}_V[i] \in \mathbb{T}^{\lfloor \frac{d}{2} \rfloor};\, i, k \in \mathbb{N}_{\leq N} \right\}$$

and

$$\Phi_T = \left\{ \boldsymbol{\phi}_T[i] = \mathcal{F}^{-1} \left\{ e^{j \Theta_T (i+b)} \right\} \,\middle|\, i \in \mathbb{N}_{\leq N} \right\}$$

where $\mathbb{T}^{\lfloor \frac{d}{2} \rfloor}$ is the surface of the positive unit hypertorus embedded in $\mathbb{R}^d$, $\Theta \sim \mathcal{N}(0, \lambda^{-2})$ for a length-scale of $\lambda = 0.675$ such that $\boldsymbol{\phi}_T^\top[i]\boldsymbol{\phi}_T[i+1] \approx \frac{1}{3}$, and $b \ggg 1$. The bias term in this case is necessary to prevent any of the vectors in Φ_T from being significantly similar to the HRR identity vector. Allowing the identity vector to be a vector within Φ_T creates a problem when dealing with bound and/or bundled vectors. As an example, consider a set of list items bundled with the associations between list items and positions:

$$\boldsymbol{u} = \sum_{i=1}^{N} \boldsymbol{\phi}_V[i] + \boldsymbol{\phi}_V[i] \circledast \boldsymbol{\phi}_T[i]$$

If we were interested in what the position of the i^{th} item of the list were, we would query $\boldsymbol{u}$ with the inverse of $\boldsymbol{\phi}_V[i]$; however, $\boldsymbol{u} \circledast \boldsymbol{\phi}_V^{\sim}[i]$ will produce a bundle containing the HRR identity vector and the i^{th} positional vector: $\boldsymbol{\phi}_{\mathbb{I}} + \boldsymbol{\phi}_T[i]$. If $\boldsymbol{\phi}_T[j] = \boldsymbol{\phi}_{\mathbb{I}} \in \Phi_T$, recall processes will be biased toward recalling the same $\boldsymbol{\phi}_T[j]$ for any list item query.

2.3.2 Spatial Memory Task

In the spatial memory task, participants navigate a room and visit a set of M distinct landmarks. We represent each landmark as a unitary SP formed by binding different combinations of colour and form SPs. Let the set of colour and form SPs be Φ_C and Φ_F respectively where c_i and f_i denote the i^{th} colours and forms to be embedded. Since we aim to represent M landmarks, we require at least $m = \left\lceil \sqrt{M} \right\rceil$ colours and forms where $\lceil \cdot \rceil$ is the ceiling function. For the i^{th} landmark, the corresponding SP can be written:

$$\boldsymbol{\phi}_L[i] = \boldsymbol{\phi}_C \left[c_{\lfloor \frac{i+1}{m} \rfloor} \right] \circledast \boldsymbol{\phi}_F \left[f_{\text{mod}(i,m)+1} \right] \tag{2.3}$$

such that each $\boldsymbol{\phi}_L[i]$ is a unique combination of colour and form vectors. To represent the navigable space in which each landmark is placed, we use a HexSSP embedding (See Section 2.2.1). The use of a HexSSP embedding was motivated by the HexSSP's underlying biological plausibility as well as the radial basis kernel that HexSSPs induce, creating a more cohesive representation of the two-dimensional space (Dumont & Eliasmith, 2020). We can write both the landmark set and the navigable space as:

$$\Phi_L = \{ \boldsymbol{\phi}_L[i] := \text{per Equation 2.3} \,|\, i \in \mathbb{N}_{\leq M} \}$$

and

$$\Phi_S = \left\{ \boldsymbol{\phi}_S(\boldsymbol{x}) = \mathcal{F}^{-1} \left\{ e^{j\Theta \boldsymbol{x}} \right\} \middle| \, \boldsymbol{x} \in \mathbb{R}^2 \right\}$$

where Θ is built according to the process laid out in Dumont and Eliasmith (2020).

2.4 Neural Realization

To validate our models and their use of the previously described vector representations, we require a means of testing their suitability to neural mechanisms. The Neural Engineering Framework (NEF; Eliasmith and Anderson, 2003) provides general principles for implementing models in a spiking neural network (SNN). Upon doing so, we can test and analyze neuro-computational hypotheses, gaining critical insight into the computational underpinnings of cognition.

Under the NEF, an arbitrary, time-varying, and high-dimensional signal, $\boldsymbol{\phi}(t) \in \mathbb{R}^d$, can be encoded into the activity of N neurons by modelling the i^{th} neuron's input current as:

$$J_i(\boldsymbol{\phi}(t)) = \alpha_i \langle \boldsymbol{e}_i, \boldsymbol{\phi}(t) \rangle + J_i^{\text{bias}}$$

where α is the gain scaling factor, $\boldsymbol{e}_i$ is the neuron's encoder—effectively a preferred firing direction over the feature space $\boldsymbol{x}$, and J^{bias} is the neural background current. We use the Leaky Integrate-and-Fire (LIF) neuron model to non-linearly translate input current into a spike train. The spiking rate activity of a neuron can be approximated as:

$$a_i(\boldsymbol{\phi}(t)) = G\left[J_i(\boldsymbol{\phi}(t))\right]$$

for the LIF model, the rate activity function is:

$$G\left[J_i(\boldsymbol{\phi}(t))\right] = \left(\tau_{\text{ref}} - \tau_{\text{RC}} \ln\left(1 - \frac{1}{J_i(\boldsymbol{\phi}(t))}\right)\right)^{-1}$$

where τ_{ref} and τ_{RC} are neurons' refractory and RC time constants. From the neural population's activities, the original vector can be recovered using linear decoding:

$$\hat{\boldsymbol{\phi}}(t) = \sum_{i=1}^{N} a_i(\boldsymbol{\phi}(t)) \boldsymbol{d}_i$$

where $\boldsymbol{d} \in \mathbb{R}^d$ is the neuron's decoder. Neuron decoders can be learned online or directly optimized to perform a function on $\boldsymbol{\phi}$, $f(\boldsymbol{\phi})$, using the connection weights between neural populations. The NEF also specifies how to realize dynamical systems in a single recurrently connected neural population.

Considering some linear dynamical system in $\mathbb{R}^d$ where $\boldsymbol{x}$ represents the value encoded in the neural population:

$$\dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) + B\boldsymbol{\phi}(t)$$

where $A, B \in \mathbb{R}^{d \times d}$. Using a recurrent connectivity matrix of $\tilde{A}$ and an input connection weight matrix of $\tilde{B}$ with a synaptic time constant of τ on both connections, the NEF rewrites $\boldsymbol{\phi}(t)$ as $\tilde{A}\boldsymbol{x}(t) + \tilde{B}\boldsymbol{u}(t)$. Critically, we map from linear system dynamics to neural dynamics with the identities:

$$\begin{aligned} \tilde{A} &= \tau A + I_d \\ \tilde{B} &= \tau B \end{aligned} \tag{2.4}$$

Using these identities, we map from the models presented throughout this thesis into large-scale spiking neural models that realize dynamical systems in recurrent population and functions between populations.

2.4.1 Neural Architectural Diagrams

Throughout this paper, we provide high-level architectural diagrams for the neural circuits that make up our model. Figure 2.1 provides an overview of the components within the architectural diagrams. Network diagram nodes can be used to represent neuron populations, subnetworks, or pass-through nodes. Pass-through nodes act as connection points for signals to enter or exit embedded neural networks but maintain no value themselves. Subnetworks are simple functional components that can be built from a small amount of primitive components, whereas embedded networks are more complex and will be used to simplify integration of developed models into a larger model for task simulation. Signal connections can have specific transformations or function-defined transformations, as Nengo will calculate the corresponding weight matrix using least-squares regression. Inhibitory connections prevent neural activity within a population or subnetwork. If the signal on an inhibitory connection is equal to 1, then attached populations and subnetworks will be inhibited. An inhibition inverter is used to swap between two coordinated inhibitory states; if the input to an inhibition inverter is 1, then the upstream and downstream inhibitory connections are active and inactive respectively (and vice versa for an input of 0).

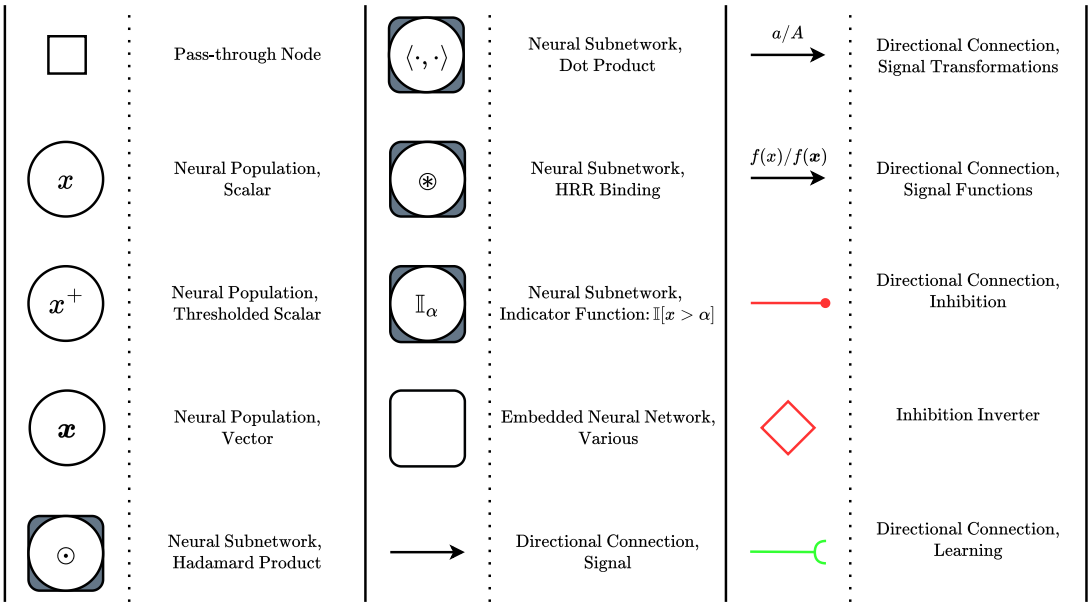


Figure 2.1: *An overview of symbols used in architectural diagrams.* Signal scaling and functions are implemented within the weights between neural populations using the NEF. Neural subnetworks build more complex functional behaviours by building neural circuits from primitives.

Chapter 3

Related Work

There has been no lack of memory modelling efforts in the past. In this chapter, we explore pre-existing analytical and neural models of memory. Mathematical models of memory are models that are implemented symbolically with or without considerations for neural plausibility. By contrast, neural models of memory are models that are implemented in neural networks. Neural models can be implemented using traditional artificial neural networks or spiking neural networks. Through exploring previous models, we highlight gaps in the existing modelling literature and consider paths forward.

3.1 Mathematical Models

3.1.1 Ordinal Serial Encoding Model

The Ordinal Serial Encoding model (OSE; Choo, 2010) is an SPA-based model of serial and free list recall that uses two vectors to act as short-term and episodic memory stores, $\boldsymbol{m}_s \in \mathbb{R}^d$ and $\boldsymbol{m}_e \in \mathbb{R}^d$ respectively. When new items are presented, the OSE degrades $\boldsymbol{m}_s$ and strengthens $\boldsymbol{m}_e$ before adding the newest item. Each store can be written as a finite geometric sum of their N items:

$$\boldsymbol{m}_s = \sum_{i=1}^{N} \gamma^{N-i} \boldsymbol{\psi}_i \quad \text{and} \quad \boldsymbol{m}_e = \sum_{i=1}^{N} \rho^{N-i} \boldsymbol{\psi}_i \tag{3.1}$$

where $\gamma \in \mathbb{R}^+_{<1}$ is the decay parameter, $\rho \in \mathbb{R}^+_{>1}$ is the strengthening parameter, and $\boldsymbol{\psi} \in \mathbb{R}^d$ is an input item represented as an SP. The overall memory of the OSE is the sum of the two stores, and items in memory can be queried by using dot product similarities

or by binding the memory with the inverse of some query vector (e.g., $\boldsymbol{m}^\top \boldsymbol{\psi}$ or $\boldsymbol{m} \circledast \boldsymbol{\psi}^\sim$ where $\boldsymbol{m} = \boldsymbol{m}_s + \boldsymbol{m}_e$).

One weakness of the OSE is the unbounded magnitude of $\boldsymbol{m}_e$. While the short-term memory store degrades older items and will eventually reach some maximum magnitude, the episodic memory store's magnitude increases by a factor of ρ for every newly presented item. Consequently, the episodic memory will be over-represented in $\boldsymbol{m}$ compared to the short-term memory. Secondly, the OSE has no clear consideration of time—updates to the short-term and episodic memory stores happen only when a new item is presented. Under this construction, the way $\boldsymbol{m}$ changes for two items presented hours apart is identical to how it would change for two items presented seconds apart.

3.1.2 SSP-Attend

The SSP-Attend model of visual working memory presented in Penzkofer et al. (2025) uses a similar structure to the OSE; however, it only uses the short-term memory store. Unlike the OSE, SSP-Attend uses decay based on item presentation timings as determined by human-like scan-paths of a colour wheel memory task:

$$\boldsymbol{m} = \sum_{i=1}^{N} \delta(i) \left(\boldsymbol{\phi}_c(c_i) \circledast \boldsymbol{\phi}_{x,y}(x_i, y_i) \right)$$

where $\delta(\cdot)$ defines the decay function for each i, $\boldsymbol{\phi}_c(\cdot)$ is an SSP embedding of colour on a colour wheel, and $\boldsymbol{\phi}_{x,y}(\cdot, \cdot)$ is an SSP embedding of the (x, y) position of the colour stimulus on the screen. The decay function for SSP-Attend can be written as $\delta(i) = \tilde{\gamma}^{-1}\gamma^{\tau_i}$ where τ_i is equal to the i^{th} item's timestamp divided by the average timestamp and $\tilde{\gamma}$ is the maximum value of γ^{τ_i} within a given trial such that $\delta(i) \in [0, 1]$. SSP-Attend demonstrates human-like distributions of target and non-target responses on colour wheel tasks, highlighting its efficacy in modelling visual working memory. While a step in the right direction, the SSP-Attend decay function depends on *a priori* knowledge of the average timestamp. By comparison, the SSP-Decay model, which is also presented in Penzkofer et al. (2025), uses an exponential decay function akin to that of the OSE, removing the need for average encoding time as well as the temporal aspects of decay. While the formation of Equation 3.2 is not intrinsically limited to these two decay functions, it remains unclear as to how time-dependent decay in stimulus presentation can be reliably implemented without *a priori* knowledge of stimulus timings. Further, SSP-Attend and SSP-Decay representations are temporally stable after presentation, as temporal dynamics are not a consideration within the model beyond stimulus encoding. Since the experiment presented in Penzkofer et al.

(2025) operates on a significantly smaller timescale than most working memory tasks, this may be an acceptable simplification; however, it would be worthwhile to explore the effects of post-presentation temporal decay on the distribution of target and non-target model responses.

3.1.3 Non-Associative Additive Model

The Non-Associative Additive model (NAA; Reimann, 2025) is a non-SPA, mathematical model of working memory that proposes the use of a non-associative vector-symbolic algebra. Similar to the OSE, the NAA maintains working memory across two distinct memory stores, $\boldsymbol{R}$ and $\boldsymbol{L}$ accounting for primacy and recency effects respectively. For the NAA, stimuli are d-dimensional binary vectors, $\boldsymbol{v} \in \{0,1\}^d$, that can be entangled using bundling and binding. The NAA's bundling operator, $+_p$, probabilistically combines two vectors according to:

$$[\boldsymbol{v}_1 +_p \boldsymbol{v}_2]_i = \begin{cases} [\boldsymbol{v}_1]_i \text{ AND } [\boldsymbol{v}_2]_i, & x_i \leq p \\ [\boldsymbol{v}_1]_i \text{ OR } [\boldsymbol{v}_2]_i, & x_i > p \end{cases}$$

where $x_i \sim \mathcal{U}(0,1)$ is a random uniform value for each index that determines its bundling behaviour, $p \in [0,1]$ is the probability of the "AND" operation for a single index's bundling, and $[\cdot]_i$ is the i^{th} entry of a vector. By its construction, $(\boldsymbol{v}_1 +_p \boldsymbol{v}_2) +_p \boldsymbol{v}_3 \neq \boldsymbol{v}_1 +_p (\boldsymbol{v}_2 +_p \boldsymbol{v}_3)$. $\boldsymbol{R}$ and $\boldsymbol{L}$ are constructed by performing bundling operations in opposite orders:

$$\boldsymbol{R} = \boldsymbol{v}_1 +_p (\boldsymbol{v}_2 +_p (\ldots +_p (\boldsymbol{v}_{N-1} +_p \boldsymbol{v}_N) \ldots))$$
$$\boldsymbol{L} = ((\ldots (\boldsymbol{v}_1 +_p \boldsymbol{v}_2) +_p \ldots) +_p \boldsymbol{v}_{N-1}) +_p \boldsymbol{v}_N$$

The NAA is capable of demonstrating human-like performance in list recall tasks, showing recency and primacy effects consistent with human behavioural data. One caveat of the NAA is that $\boldsymbol{R}$ cannot be constructed without full knowledge of a sequence, as it requires bundling beginning with the latest items of the sequence. Reimann argues that this can be thought of as "chunking", where one decomposes a sequence of N items into a sequence of $M < N$ chunks (e.g., a sequence of 10 digits being decomposed into a sequence of 5 two-digit numbers). While this is an interesting account of chunk formation, it creates a problematic recursive relationship for chunking where $\boldsymbol{R}$ still depends on full knowledge of the sequence of chunks for construction.

3.1.4 Temporal Context Model

Tensor products are commonly used for building associations between vectors, enabling nested, structured representations in higher order tensors. Smolensky (1990) demonstrates how tensor products can be used to create and navigate compositional structures. As previously mentioned, nested associations using tensor products result in an explosion of tensor order; however, the HRRs provide an alternative formulation using a compression of the tensor product. By consequence, HRR binding can be understood as a form of tensor product association. Many existing models of memory use tensor products for associations, such as the Temporal Context Model (TCM; Howard and Kahana, 2002). In the TCM, memories are formed by successive associative processes that map features to contexts and contexts to features. Contexts in this model are internally-generated from a random vector that stochastically updates based on the input features.

The association matrix of the TCM that maps from contexts to features, M^{CF}, can be written as a summation of the tensor products between feature and context vectors:

$$M^{CF} = \sum_{i=1}^{N} \boldsymbol{f}_i \boldsymbol{c}_i^{\top}$$

where N is the number of items presented within a task and all feature vectors, $\boldsymbol{f}_i \in \mathbb{R}^d$, and context vectors, $\boldsymbol{c}_i$ are approximately orthogonal. The process by which the internal context vector updates can be written as:

$$\boldsymbol{c}_i = \rho \boldsymbol{c}_{i-1} + \beta \tilde{\boldsymbol{c}}_i$$

where $\beta \in \mathbb{R}^{+}_{\leq 1}$ is the drift parameter, $\rho = \sqrt{1 - \beta^2}$ is the counter-balancing drift parameter under the assumption that items are not presented for a significant amount of time, $\tilde{\boldsymbol{c}}_i$ is the initial context estimate, and $\boldsymbol{c}_{i-1}$ is the previous context vector state. Features are recovered from a context vector with an additional association matrix, M^{FC}. Updating M^{FC} is a slightly more complicated process than updating M^{CF}; however, both matrices are effectively sums of tensor products.

The TCM demonstrates human-like list recall behaviours for free recall paradigms, as the internal context state can drive item recall. Using the association matrices to embed the relationships between contexts and features allows for a more neurally-plausible account, as these matrices can be thought of as connection transforms or weights between neural populations. Similar to the previously described models of working memory, the update processes within the TCM only occurs when a new feature vector is presented; it lacks temporal dynamics and instead substitutes it with the contextual drift process.

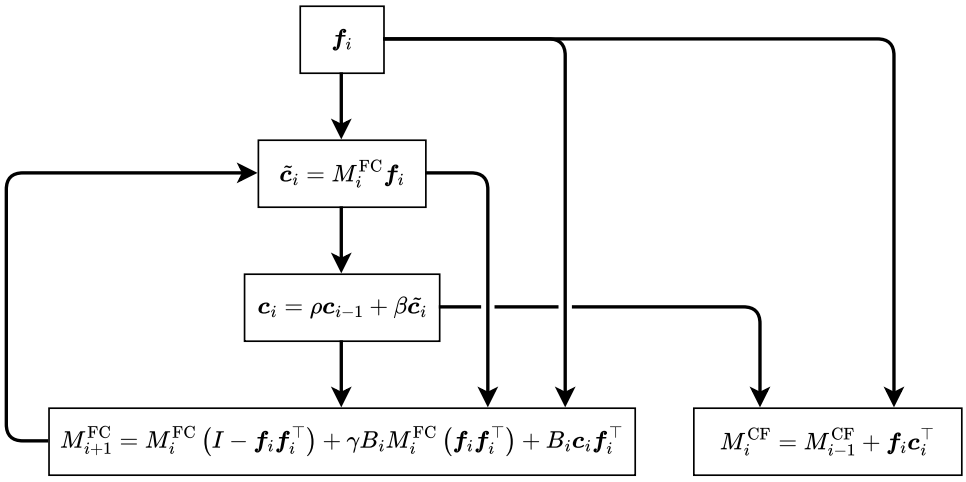


Figure 3.1: *A diagram showing the update processes of the Temporal Context Model.* The presentation of the i^{th} feature vector, $\boldsymbol{f}_i$, triggers a stochastic update to the context vector, $\boldsymbol{c}_i$, based on previous and estimated context vectors, $\boldsymbol{c}_{i-1}$ and $\tilde{\boldsymbol{c}}_i$ respectively. The feature vector, the updated context vector, and the estimated context vector are then all used to update the feature-to-context association matrix for the next feature vector, M_{i+1}^{FC} where B_i is a function of $\tilde{\boldsymbol{c}}_i^\top \boldsymbol{c}_i$ and γ—a free parameter determining the relative contribution of the second and third terms of the M_i^{FC} update equation. The updated context and feature vector are also used to update the context-to-feature mapping, M_i^{CF}, highlighting the dependence of ordered operations in updating the model's internal variables.

Further, the TCM requires feature vectors to be orthogonal and for updates to contexts and association matrices to follow some specific order. While this is trivial to implement in a traditional computing framework where every operation is performed successively, it is unclear how the mathematical basis of the TCM could be extended to a parallelized dynamical system in which these operations happen continuously over time, i.e., a spiking neural system like the brain. Figure 3.1 shows the update steps within the TCM, showing sequential dependence with directional arrows. We note that there is significant sequential dependence between association matrix update processes and context update processes. If all of these updates were occurring simultaneously, we would expect significant degradation of the TCM model's performance, due to the lack of coordination between context and association matrix updates—the primary driver in its ability to form forward contextual associations and to perform recall.

3.1.5 Context Maintenance and Retrieval Model

The Context Maintenance and Retrieval model (CMR model; Polyn et al., 2009) generalizes the TCM by introducing additional flexibility—balancing "temporal" and experimental contributions to learning new context-to-feature mappings. The structure of the association matrices for the CMR model is highly similar to that of the TCM—notably with additional scaling parameters in the learning process of M^{CF}. In doing so, the CMR model extends the TCM's free recall abilities by introducing human-like grouping behaviour for recalled items, creating a model with increasingly human-like performance. However, the CMR does little to address some of the previously mentioned weaknesses of the TCM. In particular, the CMR similarly depends on specific ordering for updating contexts and association matrices (see Figure 3.1); these updates are simple in a traditional, sequential computing framework, but it is unclear how the brain might implement something like this in a parallelized dynamical system.

3.2 Artificial Neural Models

3.2.1 Tolman-Eichenbaum Machine

The Tolman-Eichenbaum Machine (TEM; Whittington et al., 2020) is a neural model of weight-based memory based on predictive-coding principles. The TEM uses Hebbian learning to update an internal predictive model based on stimulus vectors and task-dependent context vectors. In doing so, the TEM builds an internal model of the task environment that associates task contexts to sets of stimuli and actions. Between Hebbian learning episodes, the entirety of the TEM is slowly trained using backpropagation as a substitute for hippocampal consolidation processes. Offline training allows the TEM to generalize, to some extent, across tasks. The TEM was tested on learning and completing graph-based tasks, including semantic and spatial graphs; however, the TEM requires a finite set of one-hot vectors that describe possible actions at each point in the graph. While the TEM is implemented using an artificial neural network, it uses both backpropagation and rectified linear unit (ReLU) activation functions—characteristics inconsistent with biological neurons.

3.2.2 Modern Hopfield Networks

Modern Hopfield Networks (MHNs; Ramsauer et al., 2020) are a class of recurrently-connected neural models of associative memory where weights allow for reconstruction

of patterns from partial inputs. In contrast to traditional Hopfield Networks and tensor product matrices, MHNs allow for exponential storage capacity with respect to input dimensionality. Previous models of episodic memory and recall have used MHNs as a basis for memory construction and recollection, such as the generative model of memory construction and consolidation (GMCC; Spens and Burgess, 2024) where an MHN tokens auto-associative memory traces and variational auto-encoders maintain longer-term memories in latent space. Further, a previous spiking implementation of an MHN using a pseudo-softmax operation demonstrated an increase in biological plausibility (Snow & Orchard, 2022); however, it was limited to binary feature vectors rather than continuous feature spaces and was not trained online.

3.2.3 Entity Network

An Entity Network (EntNet; Henaff et al., 2016) is a recurrent neural model of memory where memory slots roughly correspond to isolated association processes that map from keys to values. The state of the model is then updated based on the outputs of the slots using recurrent neural connections. EntNets have demonstrated exceptional sequence-processing and sequence generation abilities; however, they are limited in their applicability. Further, they have limited biological feasibility. Similar to the TEM's finite action vectors, the EntNets have a finite set of entity gates whose associations are developed using backpropagation. Due to its recurrent connectivity, the network must be unrolled into a series of sequential, feed-forward networks to perform backpropagation, creating additional problems from the lens of biological plausibility.

3.3 Spiking Neural Models

3.3.1 Spiking Neural Ordinal Serial Encoding Model

The spiking neural Ordinal Serial Encoding model (snOSE; Choo, 2010) is an implementation of the OSE in a spiking neural network using the NEF. By constructing multiple successive signal-gated memory stores with long-range recurrence, the snOSE realizes Equation 3.1 in spiking neurons. The snOSE demonstrates human-like recall abilities in both serial and free recall experiments with pronounced primacy and recency effects. Further, the snOSE has been used within other large-scale spiking neural models, such as the Semantic Pointer Architecture Unified Network (SPAUN; Eliasmith et al., 2012), to implement distributed working memory modules across modalities and tasks. In doing so, distributed

instances of the snOSE demonstrate a human-like ability to maintain, process, and recall memories in a neurally-plausible way. One weakness of the snOSE is the treatment of the episodic memory as an activity-based buffer, whereas neuroscientific findings demonstrate that the formation of episodic memories depends largely on synaptic weight changes. An additional weakness comes from the structure of the memory stores—they are designed to act as neural flip-flops with orchestrated, successive update processes.

3.3.2 Context Unified Encoding Model

The Context Unified Encoding model (CUE; Gosmann and Eliasmith, 2021) is a spiking neural working memory model that incorporates the short-term memory component of the snOSE with a Hebbian implementation of the TCM. CUE embeds the association matrices of the TCM into the decoders between neural populations and updates them based on a dynamic approximation of the TCM's M^{CF} and M^{FC} update rules:

$$\dot{D} = \eta \boldsymbol{v}(t)\boldsymbol{u}(t)^\top D\mathbb{I}\left[\langle \boldsymbol{v}(t), \hat{\boldsymbol{v}}(t)\rangle \leq \alpha\right]$$

where $\eta \in \mathbb{R}^+$ is the learning rate, $\boldsymbol{u}(t) \in \mathbb{R}^d$ is the time-varying input vector, $\boldsymbol{v}(t) \in \mathbb{R}^d$ is the time-varying target vector, $D \in \mathbb{R}^{d\times d}$ is the current decoder matrix, $\alpha \in \mathbb{R}^+$ is the similarity learning threshold, and $\mathbb{I}[\cdot]$ is an indicator function to prevent the target vector approximation, $\hat{\boldsymbol{v}}(t) \in \mathbb{R}^d$, from exceeding the similarity threshold. The previously described Hebbian learning rule is called the Association Matrix Learning (AML) rule, and it enables learning of vector-to-vector mappings using high-dimensional representations and spiking neurons. The CUE model suffers from similar limitations as the TCM and CMR models where context updates and learning processes require precise orchestration. Further, the CUE model scales the AML rule's learning threshold using an exponential decay where an *a priori* time constant determines the nature of the primacy effect.

3.4 Summary

Throughout the previous sections, we have discussed various previous modelling attempts and highlighted their shortcomings. We note that nearly every short-term memory model treats memory as a discrete process (e.g., OSE, SSP-Attend, CUE, and NAA models); changes to values within these models are limited to updates as new vectors are presented. Similarly, many models do not distinguish between short-term, intermediate-term, or long-term memory (e.g., TCM, CMR, MHN, and EntNet models), creating a gap between model explainability and human-like memory. Artificial neural network models further tend to use

activation functions inconsistent with biological neuron activities—communicating using rate-like approximations rather than spiking, neuron-like activities. Lastly, traditional models typically use backpropagation (e.g., EntNet and TEM models) or stochastic update processes (e.g., TCM, CMR, CUE, NAA models) for learning network weights—the former of which is not biologically plausible and the latter of which has limited feasibility due to the orchestration of update processes. The lack of biological plausibility in these models presents an explainability problem: might the brain actually produce the target behaviours in absence of these mechanisms? We summarize the explored shortcomings as follows: a lack of consideration for temporal dynamics; a lack of biological plausibility in form and/or function; and a lack of explainability—particularly as it concerns the mechanisms underlying learning and forgetting.

Chapter 4

Short-term Memory Modelling

The short-term memory (STM) model is one half of our overall memory model. STM is commonly characterized as transient and limited in its capacity; it is the product of neural activities that change dynamically as attention shifts throughout tasks. In creating a model of STM, we aim to replicate these key characterizations. In lieu of modelling a complete perceptual machine to convert stimuli to neural representations—i.e., an entire perceptual system—we use neurally feasible representations, Semantic Pointers (see Chapter 2), as model inputs. By using SPs, we open the door to constructing semi-symbolic dynamical systems.

4.1 Temporal Dynamics

As a first step towards realizing a continuous-time characterization of memory in which temporal dynamics play a key role, we consider a high-dimensional, leaky, and integrating dynamical system:

$$\dot{\boldsymbol{m}}(t) = \eta \boldsymbol{m}(t) + \boldsymbol{\psi}(t) \tag{4.1}$$

where $\boldsymbol{m}, \boldsymbol{\psi} \in \mathbb{R}^d$ and $\eta \in \mathbb{R}^-$. For brevity, we exclude the "(t)" moving forward and recognize that $\boldsymbol{m}$ and $\boldsymbol{\psi}$ in this context are time-varying signals unless otherwise noted. Since $\eta < 0$, Equation 4.1 realizes an exponentially-decaying system with a time constant of η^{-1}. The frequency-space transfer function can be written as:

$$H(s) = \frac{1}{s - \eta} \tag{4.2}$$

To get an initial estimate for η, we refer to the OSE in which $\gamma = 0.9775$ (Choo, 2010). This parameter was matched to human data based on a 1-second presentation time and a

brief retention interval prior to recalling a 5-item list. Because γ is based on list items and not timing, we consider the recursive OSE formulation's discrete-time transfer function, $H[z]$. We further convert it into a continuous, frequency-space transfer function, $H(s)$, using a zero-order-hold approximation:

$$\begin{aligned}
\boldsymbol{m} = \sum_{i=1}^{N} \gamma^{N-i} \boldsymbol{\psi}[i] &\rightleftharpoons \boldsymbol{m}[i+1] = \gamma \boldsymbol{m}[i] + \psi[i] \\
\mathcal{Z}\left[\boldsymbol{m}[i+1]\right] &= \mathcal{Z}\left[\gamma \boldsymbol{m}[i] + \psi[i]\right] \\
z\boldsymbol{M}[z] &= \gamma \boldsymbol{M}[z] + \boldsymbol{\Psi}[z] \\
(z-\gamma)\boldsymbol{M}[z] &= \boldsymbol{\Psi}[z] \\
H[z] = \frac{\boldsymbol{M}[z]}{\boldsymbol{\Psi}[z]} &= \frac{1}{z-\gamma} \\
\mathrm{ZOH}(H[z]) \approx \frac{c_1}{s-c_2} &\approx H(s)
\end{aligned} \tag{4.3}$$

To calculate $H(s)$ from $H[z]$, we use the MATLAB function `d2c` (The MathWorks Inc., 2026)—a function for transforming discrete transfer functions to continuous transfer functions. We find $c_1 \approx 1.01$ and $c_2 \approx -2.28 \times 10^{-2}$. The resultant system (Equation 4.3) is structurally identical to that of a leaky integrator's transfer function (Equation 4.2). We set the value of η to that of c_2: $\eta = c_2$, and since c_1 acts as a gain on the input for the continuous-time OSE-like system, we lump this gain into a flexible gain parameter, $g \in \mathbb{R}^+$:

$$\dot{\boldsymbol{m}} = \eta \boldsymbol{m} + g\boldsymbol{\psi} \tag{4.4}$$

Despite being formed from a conversion of the OSE, the system described by Equation 4.4 includes clear dynamics that decay the memory vector as time goes on. To test this system in a neural setting, we calculate state-space matrices that realize this system's described behaviour using Equation 2.4. For the recurrent connection, $A' = (\tau\eta + 1)I_d$ where $\tau \in \mathbb{R}^+$ is the lowpass synaptic time constant of the population's recurrent connection. The input transform is $B' = \tau g I_d$. Since $\tau\eta < 0$ by definition, A is a scaled-down identity matrix with equivalently scaled-down eigenvalues and eigenvectors. Consequently, there are no stable points aside from $\mathbf{0}$. That is to say, the system converges to $\mathbf{0}$ in absence of input and in abundance of time.

Figure 4.1 compares short-term memory traces of the OSE, the snOSE, a parameter-matched leaky integrator (Equation 4.4), and a recurrently-connected neural population implementing the parameter-matched leaky integrator. These traces, as well as future traces, show how the dot product between input vectors and output vector change over

time. In this case, our input vectors are SPs, and our output vector is each system's memory representation. An eidetic memory would remember each presented item perfectly—the dot product would climb to 1 and would remain at 1 forever—whereas we would expect human-like memory to forget items over time—the dot products would climb to 1 and then begin to fall once the stimulus is taken away. For each system, we presented five SPs for one second each and allowed simulations to run for 100s. For the first six seconds of the simulation, the OSE and the leaky integrator are fairly consistent. We see a comparable amount of decay for early items as presentation goes on; however, the decay after 95s of simulation is significantly higher in the leaky integrator than in the OSE. Unlike the OSE, the snOSE demonstrates a decay-like process where the dot products between the memory representation and the input vectors gradually decrease in absence of new input vectors. Similarly, the neural leaky integrator decays less than the non-neural leaky integrator. The overall stability and decay of our neural implementations may be confounded by their approximations of the A' and B' matrices. For the OSE (a), the value only changes when new items are presented, resulting in perfectly stable representations over time. For the snOSE (b), the finite approximation of an identity matrix as its recurrent connection leads to a reduced number of stable points in the recurrent populations, leading to some representational drift. For the leaky integrator (c), the decay parameter results in a scaled-down identity matrix as its ideal recurrent connection, resulting in no stable points except $\mathbf{0}$. Lastly for neural leaky integrator (d), the approximation of the scaled-down identity matrix leads to a number of non-trivial stable points, potentially preventing weak representational decay. For both the snOSE and the neural leaky integrator, the number of stable points is based on the quality of the approximations of A'. Under the NEF, the quality of signal transformations is directly tied to the number of neurons representing the original signal, as it performs least-squares regression over decoder matrices to optimize transformations (Eliasmith & Anderson, 2003). Increasing the number of neurons would then result in behaviour more consistent with the non-neural circuits—at least with respect to representational decay due to time.

Unlike the OSE and the snOSE, the leaky integrator's behaviour is sensitive to both g and the stimulus exposure time. If g is too large or a stimulus vector is presented too long, the stimulus vector will dominate the memory being represented by the leaky integrator, causing a complete erasure of items already in memory. By contrast, the discrete treatment of items by the OSE and the snOSE limits the amount of the input vector that can be introduced into memory. Ideally, a dynamic system realizing a short-term memory would bound the memory representation such that its dot product similarity does not significantly exceed 1. Lastly, we note that there is significant degradation of the memory representations as new items are presented in both neural implementations—the nature

of which is not reflected in either system's underlying model. We identify and explore representational degradation due to neural saturation in Section 4.3.2.

4.2 Bounded Representations

In order to place a bound on the magnitude of the memory representations, we first consider ideal system behaviour. As an example, we analyse how the system defined in Equation 4.4 would respond to a constant input of the HRR identity vector, $\boldsymbol{\psi}_{\mathbb{I}} = [1, 0, 0, \ldots, 0]^\top \in \mathbb{R}^d$:

$$\dot{\boldsymbol{m}} = \eta \boldsymbol{m} + g\boldsymbol{\psi}_{\mathbb{I}} \tag{4.5}$$

There are three key observations we can make: all entries of $\boldsymbol{m}$—except the first—would stay at 0; the system would continuously lose some integrated value due to the $\eta\boldsymbol{m}$ term; and the system would indefinitely integrate the g-scaled input vector. As time goes on for Equation 4.5:

$$\lim_{t\to\infty} \boldsymbol{m} = [\infty, 0, 0, \ldots, 0]^\top$$

and

$$\lim_{t\to\infty} \langle \boldsymbol{m}, \boldsymbol{\psi}_{\mathbb{I}} \rangle = \infty$$

From a semantic perspective where a given SP, $\boldsymbol{\psi}$, corresponds to a single item or concept, it is non-obvious how to interpret similarities greater than 1.0. For the example given in Equation 4.5, we would desire the following behaviours:

$$\lim_{t\to\infty} \boldsymbol{m} = [1, 0, 0, \ldots, 0]^\top$$

and

$$\lim_{t\to\infty} \langle \boldsymbol{m}, \boldsymbol{\psi}_{\mathbb{I}} \rangle = 1$$

We refer to this behaviour as memory "definite-ness": if any vector is presented indefinitely, the value of $\boldsymbol{m}$ should converge to some finite value that reflects the presence of the vector in the memory. One way to realize this desired behaviour would be to limit the magnitude of $\boldsymbol{m}$ to 1 by increasing the decay of $\boldsymbol{m}$ once the limit is reached. We can achieve this by introducing an indicator function into the feedback dynamics of the system:

$$\dot{\boldsymbol{m}} = (\eta + \beta \mathbb{I}(\|\boldsymbol{m}\|_2 \geq 1))\boldsymbol{m}(t) + g\boldsymbol{\psi} \tag{4.6}$$

where $\beta \in \mathbb{R}^-$ and $\mathbb{I}(\cdot) = 1$ for a "true" expression and 0 otherwise. The value of β can be related to η and g by evaluating the system's stability when the memory is exactly

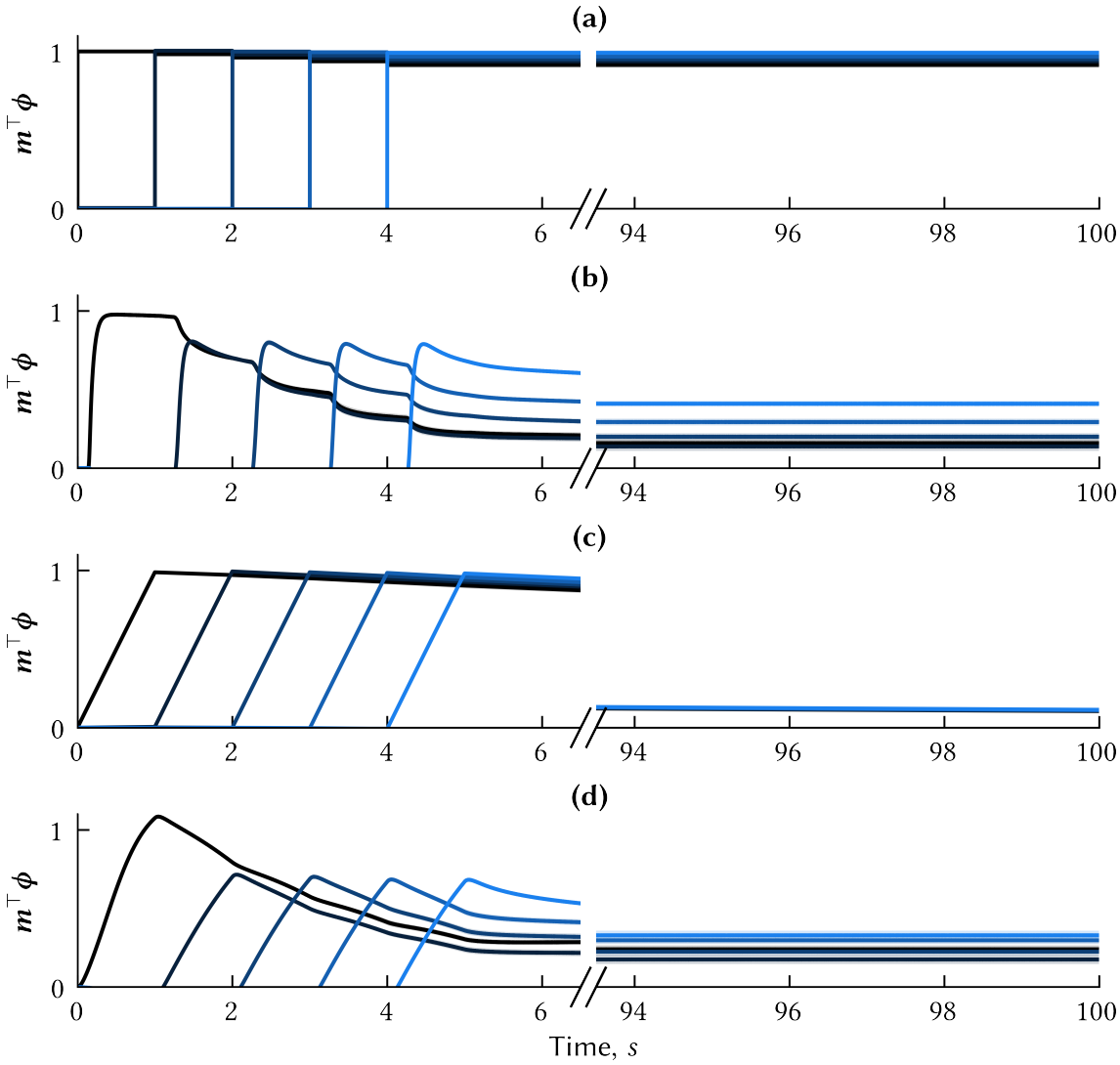


Figure 4.1: *A comparison of the OSE and snOSE to a recurrent dynamic system.* Each row shows the first and last 6 seconds of each model implementation averaged over 50 random trials. Shaded regions denote 95% confidence intervals. **(a)**: the OSE ($d = 128$, $\gamma = 0.9775$), **(b)**: the snOSE ($d = 128$, $\gamma = 0.9775$, $\tau = 0.1$, and 150 neurons per dimension per population), **(c)**: the system defined by Equation 4.4 ($d = 128$, $\eta = -2.28 \times 10^{-2}$, $g = 1$), and **(d)**: a spiking neural implementation of the system defined by Equation 4.4 ($d = 128$, $\eta = -2.28 \times 10^{-2}$, $g = 1$, $\tau = 0.1$, and 150 neurons per dimension per population).

magnitude 1. Of note, we still wish for the integrator to be leaky such that when the input is removed, the system will be driven towards $\mathbf{0}$—i.e., $\dot{\boldsymbol{m}} = \gamma \boldsymbol{m}$:

$$\eta \boldsymbol{m} = (\eta + \beta)\boldsymbol{m} + g\boldsymbol{\psi}$$

and since the system will be driven towards matching the value of $\boldsymbol{m}$ to the value of $\boldsymbol{\psi}$:

$$\begin{aligned} \eta\boldsymbol{\psi} &= (\eta + \beta)\boldsymbol{\psi} + g\boldsymbol{\psi} \\ \mathbf{0} &= (\beta + g)\boldsymbol{\psi} \\ 0 &= \beta + g \\ \beta &= -g \end{aligned} \tag{4.7}$$

Substituting Equation 4.7 back into Equation 4.6:

$$\dot{\boldsymbol{m}} = (\eta - g\mathbb{I}\left[\|\boldsymbol{m}\|_2 \geq 1\right])\boldsymbol{m} + g\boldsymbol{\psi} \tag{4.8}$$

While the system defined by Equation 4.8 demonstrates memory definite-ness for $\eta \in \mathbb{R}^-$ and $g \in \mathbb{R}^+$, it introduces a significant limitation: a magnitude limit of 1 means it can represent, at most, one SP. Qualitatively, this system allows for the memory representation to move towards the input vector until their values are equal. As an alternate formation, we can use the relationship between the vector dot product and its magnitude, $\langle \boldsymbol{\psi}, \boldsymbol{\psi} \rangle = \|\boldsymbol{\psi}\|_2^2$, to rewrite the condition of the indicator function since it is true for some non-negative x that when $x \geq 1$ that $x^2 \geq 1^2$:

$$\begin{aligned} \mathbb{I}\left[\|\boldsymbol{m}\|_2 \geq 1\right] &= \mathbb{I}\left[\|\boldsymbol{m}\|_2^2 \geq 1\right] \\ &= \mathbb{I}\left[\langle \boldsymbol{m}, \boldsymbol{m} \rangle \geq 1\right] \end{aligned}$$

and since we want the value of $\boldsymbol{\psi}$ to be fully contained within $\boldsymbol{m}$, we can recondition the indicator on its presence (i.e., dot product similarity with $\boldsymbol{m}$):

$$\mathbb{I}\left[\|\boldsymbol{m}\|_2 \geq 1\right] \Rightarrow \mathbb{I}\left[\langle \boldsymbol{m}, \boldsymbol{\psi} \rangle \geq 1\right] \tag{4.9}$$

By replacing the indicator in Equation 4.8 with the reconditioned indicator in Equation 4.9, we approach a system that both maintains definite-ness and lacks intrinsic capacity restrictions:

$$\dot{\boldsymbol{m}} = (\eta - g\mathbb{I}\left[\langle \boldsymbol{m}, \boldsymbol{\psi} \rangle \geq 1\right])\boldsymbol{m} + g\boldsymbol{\psi} \tag{4.10}$$

However once the indicator condition in Equation 4.10 is satisfied, the system subtracts out everything in $\boldsymbol{m}(t)$ rather than isolating changes to the input vector. As an example,

we can consider a case in which $\boldsymbol{m}$ is matched to some vector, $\boldsymbol{\psi}_1$, and the input is changed to $\boldsymbol{\psi}_2$. Once the dot product between $\boldsymbol{m}(t)$ and $\boldsymbol{\psi}_2$ reaches 1:

$$\begin{aligned}\dot{\boldsymbol{m}} &= (\eta - g)\left(e^{\eta \Delta t}\boldsymbol{\psi}_1 + \boldsymbol{\psi_2}\right) + g\boldsymbol{\psi_2} \\ &= (\eta - g)e^{\eta \Delta t}\boldsymbol{\psi}_1 + \eta\boldsymbol{\psi}_2 \end{aligned} \tag{4.11}$$

where $\Delta t \in \mathbb{R}^+$ is the amount of time it takes after the input switches from $\boldsymbol{\psi}_1$ to $\boldsymbol{\psi}_2$ for the dot product between $\boldsymbol{m}$ and $\boldsymbol{\psi}_2$ to reach 1—determined by the values of η and g.

Since $\eta - g < 0$, Equation 4.11 also demonstrates that the system will clear previous items from its own memory once a new item is put into memory. To prevent this, we isolate the changes to $\boldsymbol{m}$ by projecting the changes onto the input vector:

$$\begin{aligned}\dot{\boldsymbol{m}} &= \eta\boldsymbol{m} - g\mathbb{I}\left[\langle \boldsymbol{m}, \boldsymbol{\psi}\rangle \geq 1\right] \text{proj}_{\boldsymbol{\psi}}\boldsymbol{m} + g\boldsymbol{\psi} \\ &= \eta\boldsymbol{m} + g(\boldsymbol{\psi} - \mathbb{I}\left[\langle \boldsymbol{m}, \boldsymbol{\psi}\rangle \geq 1\right] \text{proj}_{\boldsymbol{\psi}}\boldsymbol{m})) \\ &= \eta\boldsymbol{m} + g(\boldsymbol{\psi} - \mathbb{I}\left[\langle \boldsymbol{m}, \boldsymbol{\psi}\rangle \geq 1\right] \langle \boldsymbol{m}, \boldsymbol{\psi}\rangle\boldsymbol{\psi}) \\ &= \eta\boldsymbol{m} + g(1 - \mathbb{I}\left[\langle \boldsymbol{m}, \boldsymbol{\psi}\rangle \geq 1\right] \langle \boldsymbol{m}, \boldsymbol{\psi}\rangle)\boldsymbol{\psi} \end{aligned} \tag{4.12}$$

Equation 4.12 defines a system that demonstrates memory definite-ness as well as a theoretically indefinite capacity to remember input vectors. We note that the capacity will also be a function of the representations' dimensionalities, as if several input vectors have non-negligible dot products to one another, it may negatively affect the actual capacity of the system. One weakness of the system is the use of the indicator function, as the dynamics of the resultant system are only piecewise linear rather than globally linear. Following the application of the linear projection, the indicator function acts to prevent the dot product between $\boldsymbol{m}$ and $\boldsymbol{\psi}$ from exceeding 1. While important, this same purpose is served by the dot product weighting of the indicator function. That is, once it has exceeded 1, $1 - \langle \boldsymbol{m}, \boldsymbol{\psi}\rangle < 0$, and the system will begin removing $\boldsymbol{\psi}$ until it reaches $\langle \boldsymbol{m}, \boldsymbol{\psi}\rangle \approx 1$. We can simplify the dynamics and make the system globally linear by removing the indicator function without loss of functionality:

$$\dot{\boldsymbol{m}} = \eta\boldsymbol{m} + g(1 - \langle \boldsymbol{m}, \boldsymbol{\psi}\rangle)\boldsymbol{\psi} \tag{4.13}$$

While we assume $\|\boldsymbol{\psi}\|_2 = 1$, we note that if $\boldsymbol{\psi}$ were instead a bundle of N orthogonal SPs, the bundle would be loaded into memory until $\langle \alpha\boldsymbol{\psi}, \boldsymbol{\psi}\rangle = 1$, resulting in $\alpha N\|\boldsymbol{\psi}\|_2 = 1$ and $\alpha = N^{-1}$. Consequently, the system would effectively introduce the average of the bundle's constitutent vectors into memory. Since the input vector, $\boldsymbol{\psi}$ will not always be unit-length, we generalize Equation 4.13 by rewriting the input as the projection of the difference between the input and the memory onto the input—which can also be understood

as a projection of the input's absence in memory onto the input itself. For a unit-length $\boldsymbol{\psi}$, $1 - \langle \boldsymbol{m}, \boldsymbol{\psi} \rangle = \langle \boldsymbol{\psi}, \boldsymbol{\psi} \rangle - \langle \boldsymbol{m}, \boldsymbol{\psi} \rangle = \langle \boldsymbol{\psi} - \boldsymbol{m}, \boldsymbol{\psi} \rangle$, and $\langle \boldsymbol{\psi} - \boldsymbol{m}, \boldsymbol{\psi} \rangle \boldsymbol{\psi} = \langle \boldsymbol{\psi} - \boldsymbol{m}, \boldsymbol{\psi} \rangle \|\boldsymbol{\psi}\|_2^{-2} \boldsymbol{\psi} = \text{proj}_{\boldsymbol{\psi}}(\boldsymbol{\psi} - \boldsymbol{m})$:

$$\dot{\boldsymbol{m}} = \eta \boldsymbol{m} + g \text{proj}_{\boldsymbol{\psi}}(\boldsymbol{\psi} - \boldsymbol{m}) \tag{4.14}$$

Equation 4.14 then defines an equivalent system that will work for non-unit input vectors, such as bundles of SPs, without erasing or exploding internal values (i.e., the system will stop introducing vectors into memory once $\langle \boldsymbol{m}, \boldsymbol{\psi} \rangle \approx \|\boldsymbol{\psi}\|_2^2$). If we suppose that input vectors for a given task will be of approximately the same magnitude, then we can maintain system linearity by approximating the normalization of $\boldsymbol{\psi}$ as a constant division by the expected squared magnitude: $\text{proj}_{\boldsymbol{\psi}}(\boldsymbol{\psi} - \boldsymbol{m}) \approx c^{-1} \langle \boldsymbol{\psi}, \boldsymbol{\psi} - \boldsymbol{m} \rangle \boldsymbol{\psi}$ where $c = \mathbb{E}\left[\|\boldsymbol{\psi}\|_2^2\right]$. The input gain parameter, g, can also serve to absorb the normalizing constant, c, resulting in the following system without overt normalization:

$$\dot{\boldsymbol{m}} = \eta \boldsymbol{m} + g \left(c - \langle \boldsymbol{m}, \boldsymbol{\psi} \rangle \right) \boldsymbol{\psi}$$

Figure 4.2 shows an analytical implementation of the projection-based memory system. We note that the memory representation does not balloon in its representation of an individual vector and does demonstrate clear temporal dynamics due to its continuous decay. We also note that as each item is presented, the magnitude of the memory representation increases proportional to the newly added item. Lastly, we note that the system described in Equation 4.14 does little to dress the capacity question of short-term memory. Short-term memory is typically significantly recency-biased and is only capable of tracking a few "chunks" of information at a time. We acknowledge that the projection-based memory system does not necessarily capture these effects; however, we note that an spiking neural implementation of this system *will* capture these effects. We focus on chunk size effects and the notion of capacity in the following section, updating the underlying analytical model and comparing it to spiking neural implementations.

4.3 Capacity

Short-term memory is widely considered to be finite in its capacity. Past working memory research has characterized its capacity in terms of chunks, typically in the range of 4-7 (Brady et al., 2011; Cowan, 2001). Each chunk can be multiple individual pieces of information, such as short sequences of digits within a phone number or the colour, form, and size of some object. To address this point, we first examine what the magnitude of a memory representation should be by examining an ideal case; we then address how neural saturation allows us to achieve this without changing the neural model's architecture.

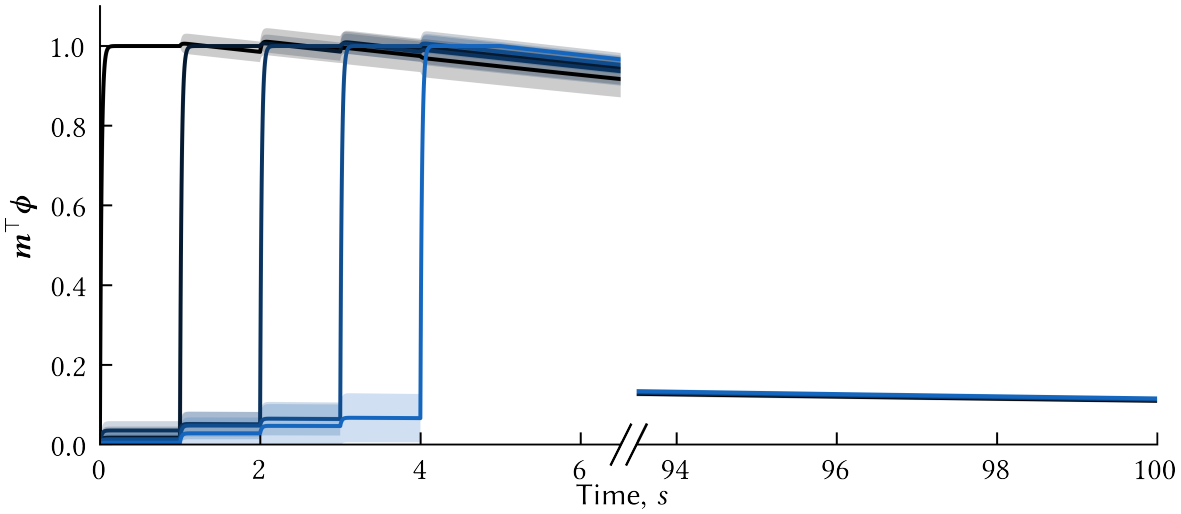


Figure 4.2: *An analytical simulation of the projection-based memory system.* Lines show dot product similarities of input vectors and the resultant memory representation for the system defined in Equation 4.14 ($d = 128$, $\eta = -2.28 \times 10^{-2}$, $g = 50$). Shaded regions show 95% confidence intervals ($N = 50$).

4.3.1 Remembering Chunks

We can represent a chunk of information as an SP or as some result of HRR operations, e.g., $\boldsymbol{c} = (\boldsymbol{\psi}_{\text{BLUE}} + \boldsymbol{\psi}_{\text{LITHE}}) \circledast \boldsymbol{\psi}_{\text{CAT}}$. In this example, we represent a lithe Russian Blue cat as a bundle of its color and its form, noting that a single chunk is often a composite of multiple pieces of information. We consider the "capacity" of the memory, $N \in \mathbb{Z}^+$, as the maximum number of approximately orthogonal chunks, $\boldsymbol{c}_i \in \mathcal{C}$ such that $\boldsymbol{c}_i^\top \boldsymbol{c}_j \approx 0 \; \forall \; i, j \in \mathbb{Z}^+_{\leq N}$ where $i \neq j$, that can be simultaneously represented in $\boldsymbol{m}$. Lastly, we consider a vector as having been loaded into memory once it is, in its entirety, within the memory (i.e., $\langle \boldsymbol{m}, \boldsymbol{\psi} \rangle \approx 1$ for $\|\boldsymbol{\psi}\|_2 = 1$).

Consider some bundle of N chunks of information, $\boldsymbol{l} = \sum_{i=1}^{N} \boldsymbol{c}_i$. Supposing $\boldsymbol{l}$ was

perfectly represented within $\boldsymbol{m}$, then $\boldsymbol{m} = \boldsymbol{l}$ and $\|\boldsymbol{m}\|_2^2 = \|\boldsymbol{l}\|_2^2$. Since $\|\boldsymbol{l}\|_2^2 = \langle \boldsymbol{l}, \boldsymbol{l} \rangle$:

$$\begin{aligned}
\|\boldsymbol{m}\|_2^2 &= \langle \boldsymbol{l}, \boldsymbol{l} \rangle \\
&= \left\langle \sum_{i=1}^{N} \boldsymbol{c}_i, \sum_{j=1}^{N} \boldsymbol{c}_j \right\rangle \\
&= \sum_{i=1}^{N} \sum_{j=1}^{N} \langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle
\end{aligned} \tag{4.15}$$

Supposing $\boldsymbol{c}_i$ have an average mutual similarity of $\zeta = \mathbb{E}\left[\langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle_{i \neq j}\right]$, we simplify Equation 4.15 by isolating the terms for equal and unequal i and j values such that dot product and average similarity measures can be exploited:

$$\begin{aligned}
\|\boldsymbol{m}\|_2^2 &= \sum_{i=1}^{N} \sum_{j=1}^{N} \langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle \delta(i-j) + \sum_{i=1}^{N} \sum_{j=1}^{N} \langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle (1 - \delta(i-j)) \\
&= N\|\boldsymbol{c}\|_2^2 + N(N-1)\zeta \\
&= \zeta N^2 + (\|\boldsymbol{c}\|_2^2 - \zeta)N \\
\|\boldsymbol{m}\|_2 &= \sqrt{\zeta N^2 + (\|\boldsymbol{c}\|_2^2 - \zeta)N}
\end{aligned} \tag{4.16}$$

where $\delta(\cdot)$ is the Kronecker delta function such that $\delta(x) = 1$ if $x = 0$ and 0 elsewhere.

When representing an SP using a population of spiking neurons, we select distributions of representational directions and biases within the unit hypersphere such that the volume of the hypersphere is uniformly representable (Voelker et al., 2017). By consequence, neuron expressivity is severely limited outside the unit hypersphere; there would be little-to-no change between neural responses for the inputs $2 \times \boldsymbol{\psi}$ and $15 \times \boldsymbol{\psi}$ due to the saturation of individual neurons' firing rates. While this is a limitation of using spiking neurons, we can leverage it to control the theoretical capacity by modifying the radius of the hypersphere, r, accordingly.

It follows that to represent $\boldsymbol{l}$ within a single neural population, we require a representational radius equal to $\|\boldsymbol{m}\|_2$—equivalently $\|\boldsymbol{l}\|_2$. As a last step, we assume that all $\boldsymbol{c}_i$ are approximately orthogonal such that $\zeta \approx 0$. We can then characterize the radius necessary to have a memory population with a capacity of N as:

$$r = \|\boldsymbol{c}\|_2 \sqrt{N} \tag{4.17}$$

By leveraging neural saturation in a coherent way, we realize "forgetting" as a consequence of finite representational capacity rather than a form of scaling or noising between

items. Figure 4.3 illustrates how capacity changes affect representational dynamics. Most importantly, an increased capacity allows for representations to maintain similarity to an increased number of vectors simultaneously.

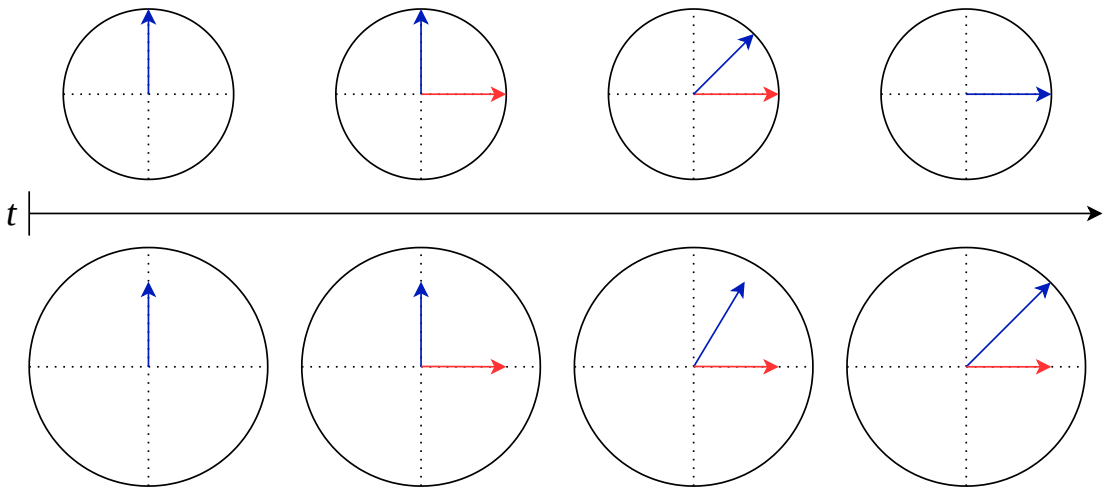


Figure 4.3: *A visualization of representational saturation in neural populations.* Representation vectors ($\boldsymbol{m}$, blue) changing due to new stimulus vectors ($\boldsymbol{\psi}$, red) for varying capacities. **Top**: A network with a capacity of $N = 1$ updating as a second vector is introduced. The representation vector rotates to match the stimulus vector, losing all similarity to the original vector. **Bottom**: A network with a capacity of $N = \sqrt{2}$ updating as a second vector is introduced. The representation vector grows in the direction of the stimulus vector due to the projection operator, retaining its similarity to the original vector.

4.3.2 Neural Saturation Dynamics

Since the neural saturation dynamics of the previous section are not realized in the dynamical system described in Equation 4.14, we aim to update the expression such that the dynamics are accounted for. Consider the behaviour demonstrated in Figure 4.3. When the magnitude of the representation is below its maximum, it can grow in the direction of the projection input, changing both in magnitude and direction; however when the magnitude is at its maximum, it rotates in the direction of the projection input and can only change in direction. Given the projection input, we can consider how the input affects the

magnitude and the direction of the representation by separating it into components that are normal ($\perp$) and tangential ($\top$) to the surface of the representational sphere at $\boldsymbol{m}$:

$$\text{proj}_{\boldsymbol{\psi}}(\boldsymbol{\psi}-\boldsymbol{m}) = \big(\text{proj}_{\boldsymbol{\psi}}(\boldsymbol{\psi}-\boldsymbol{m})\big)_{\perp} + \big(\text{proj}_{\boldsymbol{\psi}}(\boldsymbol{\psi}-\boldsymbol{m})\big)_{\top} \tag{4.18}$$

where $(\cdot)_{\perp}$ is a projection onto the unit normal vector and $(\cdot)_{\top}$ is the original vector subtracted by $(\cdot)_{\perp}$. Since a normal vector for some point,$\boldsymbol{m}$, on the surface of a hypersphere is its own unit vector, $\hat{\boldsymbol{m}} = \boldsymbol{m}\|\boldsymbol{m}\|_2^{-1}$, we can rewrite $(\cdot)_{\perp}$ and $(\cdot)_{\top}$ in Equation 4.18 as functions of projections onto $\boldsymbol{m}$:

$$\begin{aligned}\text{proj}_{\boldsymbol{\psi}}(\boldsymbol{\psi}-\boldsymbol{m}) =& \text{proj}_{\boldsymbol{m}}\left[\text{proj}_{\boldsymbol{\psi}}(\boldsymbol{\psi}-\boldsymbol{m})\right] + \\ & \text{proj}_{\boldsymbol{\psi}}(\boldsymbol{\psi}-\boldsymbol{m}) - \text{proj}_{\boldsymbol{m}}\left[\text{proj}_{\boldsymbol{\psi}}(\boldsymbol{\psi}-\boldsymbol{m})\right]\end{aligned} \tag{4.19}$$

If we were to remove the first term of Equation 4.19, we would effectively remove the component of the projection input that pushes $\boldsymbol{m}$ in a radial direction, thereby preventing changes to its magnitude. As described in the previous section and demonstrated in Figure 4.3, the lack of radial changes once the neural population saturates is key for understanding its representational capacity. To this point, we introduce a variation of the projection operator that can capture these saturation dynamics for some maximum representational radius, r:

$$\begin{aligned}\text{satproj}_r(\boldsymbol{m},\boldsymbol{\psi}) =& \nu_r(\boldsymbol{m})\text{proj}_{\boldsymbol{m}}\left[\text{proj}_{\boldsymbol{\psi}}(\boldsymbol{\psi}-\boldsymbol{m})\right] + \\ & \text{proj}_{\boldsymbol{\psi}}(\boldsymbol{\psi}-\boldsymbol{m}) - \text{proj}_{\boldsymbol{m}}\left[\text{proj}_{\boldsymbol{\psi}}(\boldsymbol{\psi}-\boldsymbol{m})\right] \\ =& \left(\nu_r(\boldsymbol{m})-1\right)\text{proj}_{\boldsymbol{m}}\left[\text{proj}_{\boldsymbol{\psi}}(\boldsymbol{\psi}-\boldsymbol{m})\right] + \text{proj}_{\boldsymbol{\psi}}(\boldsymbol{\psi}-\boldsymbol{m})\end{aligned} \tag{4.20}$$

where $\nu_r(\cdot)$ is a saturation function that discourages $\|\boldsymbol{m}\|_2$ from exceeding r. As an example, we can use an indicator function to enforce a hard threshold of $\|\boldsymbol{m}\|_2 \leq r$: $\nu_{r,\mathbb{I}}(\boldsymbol{m}) = \mathbb{I}\left[\|\boldsymbol{m}\|_2 \leq r\right]$. Alternatively, we can realize a soft threshold of $\|\boldsymbol{m}\|_2 \leq r+\epsilon$ using a smooth $\nu_r(\cdot)$, such as the logistic sigmoid function: $\nu_{r,\sigma}(\boldsymbol{m}) = \left(1+e^{k(\|\boldsymbol{m}\|_2 - r)}\right)^{-1}$ where $k \in \mathbb{R}^+$. Notably as $k \to \infty$, $\nu_{r,\mathbb{I}}(\boldsymbol{m}) \approx \nu_{r,\sigma}(\boldsymbol{m})$.

Given the reformulation of the projection input into a saturation-respecting projection input, we can rewrite the previous dynamic system formulation in Equation 4.14 using Equation 4.20:

$$\dot{\boldsymbol{m}} = \eta\boldsymbol{m} + g\text{satproj}_r(\boldsymbol{m},\boldsymbol{\psi}) \tag{4.21}$$

or more explicitly:

$$\dot{\boldsymbol{m}} = \eta\boldsymbol{m} + g\left(\left(\nu_r(\boldsymbol{m})-1\right)\left\langle\left\langle\boldsymbol{\psi}-\boldsymbol{m},\hat{\boldsymbol{\psi}}\right\rangle\hat{\boldsymbol{\psi}},\hat{\boldsymbol{m}}\right\rangle\hat{\boldsymbol{m}} + \left\langle\boldsymbol{\psi}-\boldsymbol{m},\hat{\boldsymbol{\psi}}\right\rangle\hat{\boldsymbol{\psi}}\right)$$

where $\hat{\boldsymbol{\psi}} = \boldsymbol{\psi}\|\boldsymbol{\psi}\|_2^{-1}$.

It is worth noting that the saturation dynamics expressed in Equation 4.21 are a *characterization* of the neural saturation mechanism. When the projection-based short-term memory defined in Equation 4.14 is implemented in spiking neurons, we realize the dynamics of Equation 4.21 due to the intrinsic neural dynamics. Consequently, we get the representational saturation dynamics for free. We refer to the neural circuit that implements Equation 4.14 (and due to neural saturation realizes Equation 4.21) as the STM network. Figure 4.4 shows the structure of the STM network as a block diagram where the projection operation is implemented as connected subnetworks calculating the dot product and input scaling. The maximum amount of the input to be introduced, $c^{\max}$, corresponds to the expected chunk size of the input vector. In the case of the input being a single SSP, $c^{\max} = 1$.

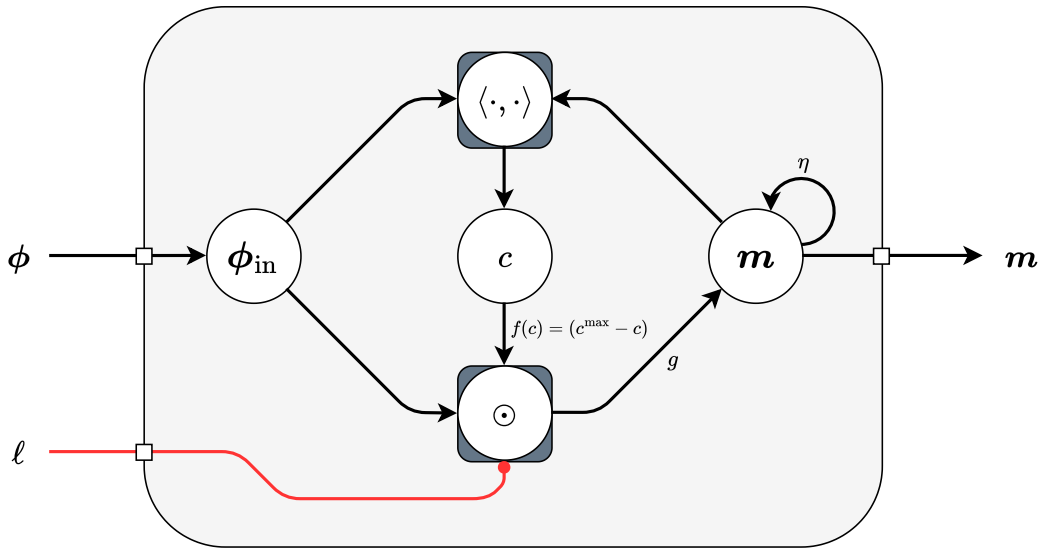


Figure 4.4: *An architectural diagram of the STM network.* Input vectors, $\boldsymbol{\phi}$, are added to the memory representation, $\boldsymbol{m}$, using the projection system described by Equation 4.14: $\text{proj}_{\boldsymbol{\phi}}(\boldsymbol{\phi} - \boldsymbol{m}) \approx (c^{\max} - \langle \boldsymbol{\phi}_{\text{in}}, \boldsymbol{m} \rangle)\boldsymbol{\phi}_{\text{in}}$ where $\boldsymbol{\phi}_{\text{in}} \approx \boldsymbol{\phi}$ and $c^{\max}$ is the expected magnitude of $\boldsymbol{\phi}$. When the inhibitory signal, ℓ, is active, input to $\boldsymbol{m}$ from the projection system drops to $\mathbf{0}$; however, the recurrent connection is unaffected. See Section 2.4.1 for an explanation of the symbols used within the architectural diagram.

Figure 4.5 shows a comparison between an analytical implementation of Equation 4.21 and the STM network implemented in spiking neurons. While the updated analytical

model does a significantly better job of capturing the behaviour of the neural model than the original analytical model (Figure 4.2), we note that remaining model mismatch could be due to the tested saturation function, $\nu_r(\cdot)$, being a good—but not perfect—fit to the neural population's saturation function. Further, results are complicated by the finite number of neurons per dimensions. Supposing a better tuned saturation function and a larger overall model, we anticipate improved similarity between the two plots; however, the current plots demonstrate significant improvement in predicting model behaviour from the system's dynamics. We note that similar to the snOSE and the spiking neural implementation of the system defined by Equation 4.4, the spiking neural implementation of Equation 4.21 (Figure 4.5(b)) demonstrates less temporal decay than the analytical system. While there is some significant decay immediately after the presentation of the last item ($t = 5$), there is limited further decay, and the resultant stable value ($t \to \infty$) if fairly far from $\mathbf{0}$. As was discussed previously, the limited number of neurons used to approximate the system dynamics can lead to the development of stable points in the representational space. This is particularly likely due to the system's weak temporal decay where realizing η requires the approximation of $\tilde{A} = \tau A + I_d$ or $\tilde{A} \approx 0.9977 I_d$ in our simulations. Since our value of η was calculated from prior work, we leave the strength of the temporal decay as a to-be-optimized parameter and continue forward with $\eta \approx -2.28 \times 10^{-2}$. For a temporal decay of strength η ($\dot{\boldsymbol{m}} = \eta \boldsymbol{m}$, we note that the exponential decay time constant is the inverse of η—$\tau_{\text{Decay}} = \eta^{-1} \approx 43.9$. Since our working memory experiments operate on time-scales near τ_{Decay} $(30 - 60s)$, an increase to the realized decay in the spiking neural implementation would require a reconfiguration of model parameters, but it would not fundamentally change the behaviour of the system as the decay would act uniformly on the memory representation.

In comparison to Figure 4.1(b) and (d), Figure 4.5(b) demonstrates more consistent loading of items into the memory representation as well similarly modest temporal decay. Further, we note that the system behaviours in Figures 4.5(a) and (b) are much more consistent than those of Figures 4.1(a) and (b) as well as Figures 4.1(c) and (d) for the portion of the simulation in which items are being presented. The behavioural consistency highlights that Equation 4.21 does a reasonably good job of describing the spiking neural system's behaviour, unlike the evident mismatch between the OSE and snOSE as well as the leaky integrator and the spiking neural leaky integrator.

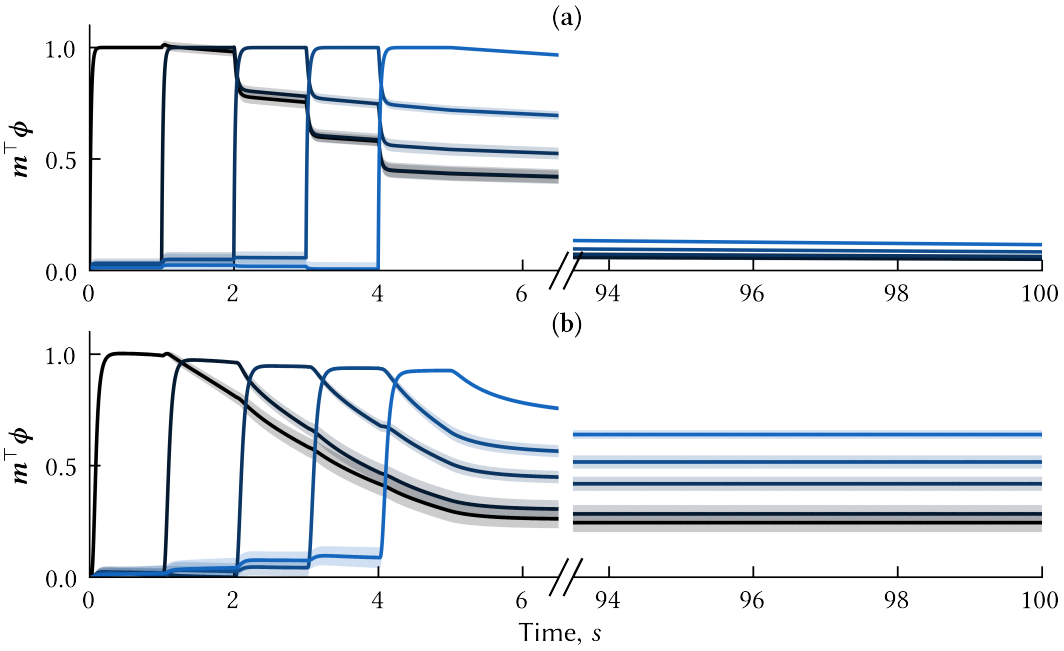


Figure 4.5: *A comparison of the analytical satproj memory model and the neural projection memory model realized in spiking neurons.* **(a)**: Dot product similarities of input vectors and the resultant memory representation for the system defined in Equation 4.21 ($d = 128$, $\eta = -2.28 \times 10^{-2}$, $g = 50$, and $\nu_{\sqrt{2},\sigma}(\boldsymbol{m})$, $k = 50$). **(b)**: Dot product similarities of input vectors and the memory representation for the system defined in Equation 4.14 implemented in spiking neurons ($d = 128$, $\eta = -2.28 \times 10^{-2}$, $g = 50$, $\tau = 0.1$, and 150 neurons per dimension per population). Shaded regions show 95% confidence intervals ($N = 50$).

Chapter 5

Intermediate-term Memory Modelling

The intermediate-term memory (ITM) model composes the second half of our overall memory model. ITM is commonly characterized as the changes in synaptic connections' strengths by which recent experiences are encoded within the brain. We differentiate ITM from long-term memory (LTM) by considering this process a form of rapid, semi-stable learning that requires consolidative processes to become stabilized and abstracted into LTM. We attribute ITM to STSP, allowing for the formation of semi-stable representations on the timescale of minutes. ITM and weight-based learning is typically associated with primacy effects observed in list recall tasks. Critically, primacy effects have been shown to decrease in significance as the amount of time between list presentation and recall increases. Consequently, our model aims to capture this effect by coupling a semi-stable learning process with an activity-based integration of task-related contexts—such as temporal sequence for a list, or spatial location for a map.

5.1 Implicit Auto-Associative Mappings

One particular Hebbian process for weight-based learning is the AML. The AML learning process prioritizes avoiding catastrophic forgetting when learning associations between high-dimensional vectors. The CUE model (see Section 3.3.2) uses the AML to approximate TCM-like association matrices; however, the CUE model also depends on precise updates to the context vectors. Since the AML is well-suited for reconstructing bundles from vectors, we seek to form the same type of association-based memory as the CUE

model without the need for a TCM-like context vector. Instead, we aim to replace the context vector with task-related contexts—e.g., the temporal embedding of the order of list items as SSPs. Since the AML forms an outer-product matrix between populations, we first consider the learned matrix, L, as:

$$L = \sum_{i=1}^{N} \boldsymbol{\phi}_{1,i} \boldsymbol{\phi}_{2,i}^{\top} \tag{5.1}$$

where $L \in \mathbb{R}^{d \times d}$ is a tensor product association matrix and $\boldsymbol{\phi}_{1,i}$ and $\boldsymbol{\phi}_{2,i}$ are arbitrary SSPs. Supposing each $\boldsymbol{\phi}_1$ is orthogonal such that $\langle \boldsymbol{\phi}_{1,i}, \boldsymbol{\phi}_{1,j} \rangle \approx 0$ for $i \neq j$ and each $\boldsymbol{\phi}_2$ is orthogonal such that $\langle \boldsymbol{\phi}_{2,i}, \boldsymbol{\phi}_{2,j} \rangle \approx 0$ for $i \neq j$, the output of $L\boldsymbol{\phi}_{2,j}$ can be written as:

$$\begin{aligned} L\boldsymbol{\phi}_{2,j} &= \left(\sum_{i=1}^{N} \boldsymbol{\phi}_{1,i} \boldsymbol{\phi}_{2,i}^{\top} \right) \boldsymbol{\phi}_{2,j} \\ &= \sum_{i=1}^{N} \left(\boldsymbol{\phi}_{1,i} \boldsymbol{\phi}_{2,i}^{\top} \boldsymbol{\phi}_{2,j} \right) \\ &= \sum_{i=1}^{N} \left(\boldsymbol{\phi}_{1,i} \left\langle \boldsymbol{\phi}_{2,i}, \boldsymbol{\phi}_{2,j} \right\rangle \right) \\ &= \boldsymbol{\phi}_{1,j} \end{aligned} \tag{5.2}$$

The tensor product association matrix allows for querying using $\boldsymbol{\phi}_{2,j}$, as demonstrated by Equation 5.2; however, if we were to attempt to query L using $\boldsymbol{\phi}_{1,j}$—e.g., $L\boldsymbol{\phi}_{1,j}$—the result would be $\mathbf{0}$ since $\langle \boldsymbol{\phi}_{2,i}, \boldsymbol{\phi}_{1,j} \rangle \approx 0$ for all combinations of i and j. Critically, this multiplication scheme is a limitation in the neural implementation—we cannot transpose and left-multiply the input and L in a way that would allow for recovery of the target vector. One option would be to create and learn a second tensor product association matrix, L', that learns the association in the opposite order. The second matrix would enable querying in the opposite direction, $L'\boldsymbol{\phi}_{2,j} = \boldsymbol{\phi}_{2,j}$; however, the maintenance of two separate tensor product association matrices is unnecessary. In fact, we can bundle the input and target vectors and treat the bundle as both the input and the target. Using the AML learning rule (see Section 3.3.2) and a sufficient similarity threshold, $\alpha = \|\boldsymbol{\phi}_{1,i} + \boldsymbol{\phi}_{2,i}\|_2^2$, we can learn a single bundle-to-bundle mapping rather than independent feature-to-context and context-to-feature mappings, as done in the CUE model. The resultant auto-associative mapping can be written in the same form as Equation 5.1:

$$L = \sum_{i=1}^{N} \left(\boldsymbol{\phi}_{1,i} + \boldsymbol{\phi}_{2,i} \right) \left(\boldsymbol{\phi}_{1,i} + \boldsymbol{\phi}_{2,i} \right)^{\top}$$

or

$$L = \sum_{i=1}^{N} \left(\left(\boldsymbol{\phi}_{1,i} + \boldsymbol{\phi}_{2,i} \right) \boldsymbol{\phi}_{1,i}^{\top} + \left(\boldsymbol{\phi}_{1,i} + \boldsymbol{\phi}_{2,i} \right) \boldsymbol{\phi}_{2,i}^{\top} \right) \tag{5.3}$$

Since Equation 5.3 describes a bundle-to-bundle mapping such that each item is associated to the bundle in its entirety, we can recover the complete bundle from only one of its constituent vectors. Supposing $\boldsymbol{\phi}_j \in \left\{ \boldsymbol{\phi}_{1,j}, \boldsymbol{\phi}_{2,j} \right\}$:

$$\begin{aligned} L\boldsymbol{\phi}_j &= \left(\sum_{i=1}^{N} \left(\left(\boldsymbol{\phi}_{1,i} + \boldsymbol{\phi}_{2,i} \right) \boldsymbol{\phi}_{1,i}^{\top} + \left(\boldsymbol{\phi}_{1,i} + \boldsymbol{\phi}_{2,i} \right) \boldsymbol{\phi}_{2,i}^{\top} \right) \right) \boldsymbol{\phi}_j \\ &= \sum_{i=1}^{N} \left(\left(\boldsymbol{\phi}_{1,i} + \boldsymbol{\phi}_{2,i} \right) \left\langle \boldsymbol{\phi}_{1,i}, \boldsymbol{\phi}_j \right\rangle + \left(\boldsymbol{\phi}_{1,i} + \boldsymbol{\phi}_{2,i} \right) \left\langle \boldsymbol{\phi}_{2,i}, \boldsymbol{\phi}_j \right\rangle \right) \\ &= \boldsymbol{\phi}_{1,j} + \boldsymbol{\phi}_{2,j} \end{aligned}$$

We refer to the network that learns online associations between components of vector bundles as the ITM_L network. Figure 5.1 shows the structure of the ITM_L network as a block diagram where the L matrix is embedded within the weights between two neural populations, $\boldsymbol{l}_{\text{pre}}$ and $\boldsymbol{l}_{\text{post}}$, while learning is carried out using the AML learning rule as described in Section 3.3.2. An additional inhibitory signal, ℓ, is used to prevent excessive learning and learning during inference time by inhibiting the target population, $\boldsymbol{t}$.

Figure 5.2 shows the behaviour of a spiking neural implementation of the ITM_L performing learning and recollection of bundle-to-bundle mappings. We present three bundles of paired orthogonal vectors, $\boldsymbol{\phi}_{\text{in},i} = \boldsymbol{\phi}_{1,i} + \boldsymbol{\phi}_{2,i}$ for $i \in \mathbb{N}_{\leq 3}$. Each pair is presented to the network for one second while learning is active, following which a single item from each pair, $\boldsymbol{\phi}_{1,i}$, is presented to query for the associated vector, $\boldsymbol{\phi}_{2,i}$. The presence of the near-1 dot product for queried vector demonstrates that the tensor association matrix, L, successfully captures the implicit association learned between the two vectors. We note that there is some variability in dot product traces during and after learning. We note that these are

The ITM_L network's reconstructive behaviour is similar to an MHN's reconstructive behaviour where an incomplete pattern can be reconstructed from an MHN's stored pattern. Unlike the STM, where capacity is understood as the number of vectors or chunks that can be represented simultaneously by the neural population, the capacity of the ITM_L is the number of reconstructive patterns it can learn. For a generic tensor product association matrix (Equation 5.1) where all vectors are approximately orthogonal, each tensor product defines a 1-dimensional subspace ($c\boldsymbol{\phi}_{2,i} \, \forall \, c \in \mathbb{R}$) in d dimensions. We can consequently

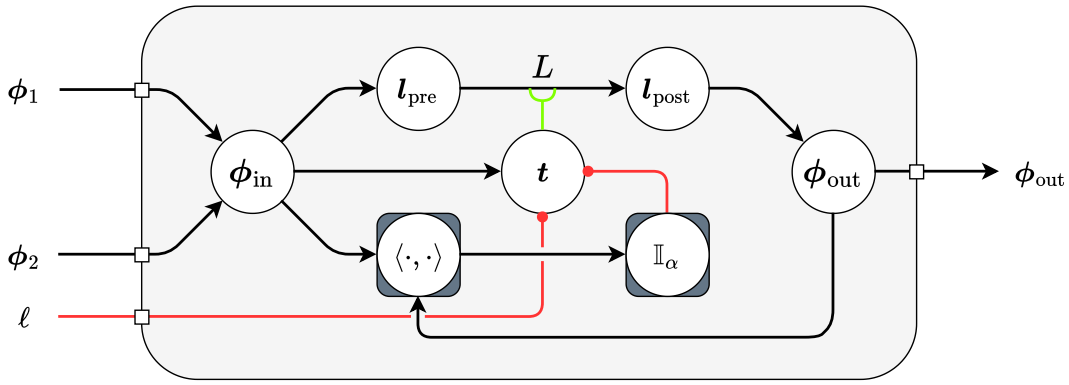


Figure 5.1: *An architectural diagram of the ITM$_L$ network.* Input vectors, $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$, are bundled, and the resultant vector, $\boldsymbol{\phi}_{\text{in}}$, is used as both the cue ($\boldsymbol{l}_{\text{pre}}$) and the target ($\boldsymbol{t}$) for the AML learning rule. The network then learns the L matrix between the pre- and post-learning populations, $\boldsymbol{l}_{\text{pre}}$ and $\boldsymbol{l}_{\text{post}}$, according to Equation 5.3. By the AML (see Section 3.3.2): $\dot{D} = \eta_L \boldsymbol{l}_{\text{pre}} \boldsymbol{t}^\top D \mathbb{I}\left[\langle \boldsymbol{l}_{\text{pre}}, \boldsymbol{t} \rangle \leq \alpha\right]$ where $\boldsymbol{\phi}_{\text{in}} \approx \boldsymbol{l}_{\text{pre}}$ and $\boldsymbol{\phi}_{\text{in}} \approx \boldsymbol{t}$ when $\boldsymbol{t}$ is not inhibited, otherwise $\boldsymbol{t} \approx \boldsymbol{0}$. See Section 2.4.1 for an explanation of symbols used within the architectural diagram.

store approximately d tensor products in L before there becomes non-negligible overlap in patterns and subspaces, leading to degradation of stored patterns. Since we extend Equation 5.1 in Equation 5.3 by using a bundle of two vectors as the bases for associations, each tensor product effectively defines a 2-dimensional subspace ($c_1\boldsymbol{\phi}_1 + c\boldsymbol{\phi}_2 \,\forall\, c_1, c_2 \in \mathbb{R}$) in the same d dimensions. While we could fit d 1-dimensional subspaces in d, we can only fit $\left\lfloor \frac{d}{2} \right\rfloor$ 2-dimensional subspaces in the same d. Consequently, the structure we use to implicitly associate these vectors halves the theoretical capacity of the network's association matrix. While this is not ideal, our high-dimensional representations still allow us to learn a sufficient number of associations to complete a variety of tasks. Ideally, we would use something more like an MHN due to its exponentially larger capacity; however, the limited neural plausibility of current MHNs make them less applicable to our current work—see Section 3.2.2 for more information.

5.2 Temporal Integration

Just as a multi-layer perceptron requires an input to have an output, the ITM$_L$ network requires prompting to recover associations ($L\boldsymbol{0} = \boldsymbol{0}$). In order to prompt the ITM$_L$ network, we consider how we might form a passive, activity-based buffer. This buffer would allow for

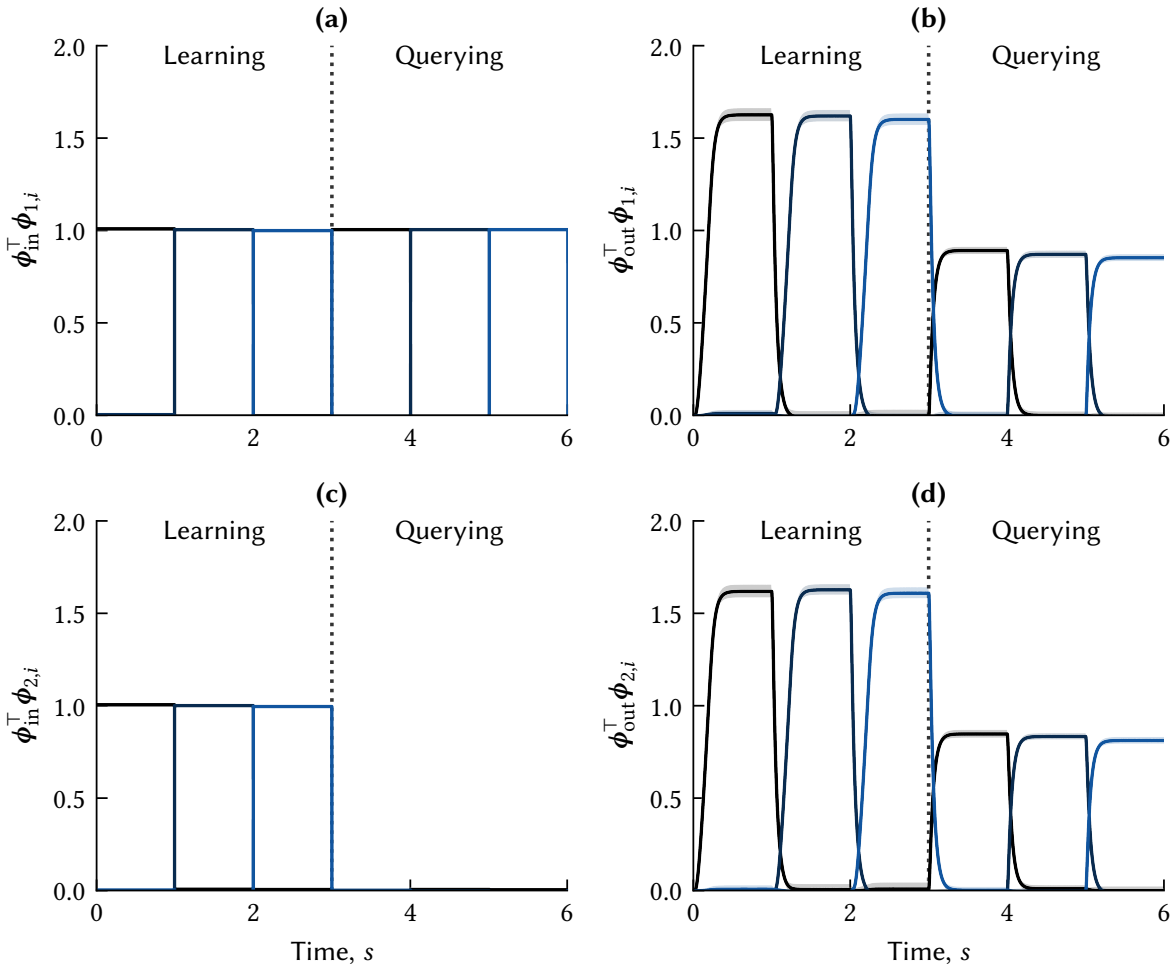


Figure 5.2: *A demonstration of learning implicit associations with the AML in spiking neurons.* Lines show dot product similarities of the input and output vectors with ground truth representations. The model was presented with $\boldsymbol{\phi}_{\text{in}}(t) = \boldsymbol{\phi}_{1,i} + \boldsymbol{\phi}_{2,i}$ where $i = \lfloor t \rfloor$ for the first three seconds, and learned the corresponding tensor association matrix. After the first three seconds, the model was queried $\boldsymbol{\phi}_{\text{in}}(t) = \boldsymbol{\phi}_{1,i}$ where $i = \text{mod}\,(\lfloor t \rfloor, 3)$ to test the ability of the learned mapping to reconstruct respective $\boldsymbol{\phi}_{2,i}$ vectors ($d = 128$, $\eta_L = 10$, $\alpha = 2$, and 250 neurons per dimension). **(a)**: The dot product of the input vector with $\boldsymbol{\phi}_{1,i}$. **(b)**: The dot product of the output vector with $\boldsymbol{\phi}_{1,i}$. **(c)**: The dot product of the input vector with $\boldsymbol{\phi}_{2,i}$. **(d)**: The dot product of the output vector with $\boldsymbol{\phi}_{2,i}$. Shaded regions show 95% confidence intervals ($N = 50$).

the model to integrate temporal information as new information is presented and to query the ITM_L with the result. Further, the associative tensor products within ITM_L network's weights have no intrinsic primacy biases, meaning that whatever activity-based buffer we develop must be capable of demonstrating some primacy bias. We construct it this way with intention: unlike the CUE model, which depends on scaling of learning processes to embed a primacy bias within the weights, we recognize that primacy effects diminish with list length, suggesting that the ability of one to situate oneself within a task depends on one's ability to maintain one's starting point and task context as a whole. As a starting point for developing the activity-based buffer, we return to the projection-based short-term memory network developed in Chapter 4. We adapt Equation 4.14 into a passive integration process by revisiting the input gain parameter and the projection operation.

To distinguish this projection-based system from the STM, we refer to the integrated representation as $\boldsymbol{p}$ and its gain parameter as g_p. To ensure a primacy effect within a neural system implementing Equation 4.14, we require the input vector to be less significant in magnitude than the recurrent vector. That is to say, for some $A' = I_d$ and $B' = g_p I_d$, a sufficiently small g_p (e.g., $g_p \leq 10$) allows for preferential maintenance of the first-presented vector due to the neural saturation mechanisms explored in Section 4.3.2. We initially selected $g_p = 2$ for its ability to provide a strong primacy effect for varying relevant list lengths—a sharp decrease from the value of $g = 50$ used for the recency effect in the STM. Since the value of g_p is so small, it becomes unable to compete with the existing memory trace, not allowing the input vector to overwrite what is currently in the memory (the dynamics of which will be explored in the following paragraphs). Further, if we filter the scaling of the projection operation to be non-negative, we prevent the system from subtracting out excess information once the dot product similarity between the input and the integrated temporal embedding exceeds 1. In doing so, these representations saturate the input and create a kind of inertia that allows the activity-based network to preferentially maintain the earliest temporal contexts within the list. We rewrite the intended dynamic system as:

$$\dot{\boldsymbol{p}} = -\eta \boldsymbol{p} + g_p f\left(\langle \boldsymbol{\phi} - \boldsymbol{p}, \boldsymbol{\phi} \rangle\right) \boldsymbol{\phi} \tag{5.4}$$

where $\boldsymbol{p}$ is the integrated task context vector representation, g_p is the altered gain parameter, $\boldsymbol{\phi}$ is the input SSP, and $f(\cdot)$ is a non-negative filtering function such that $f(x) = x$ if $x \geq 0$ and 0 otherwise. We note that this alteration also affects the characterization of the resultant system (Equation 4.21), so we provide an equivalent formation for the filtered system:

$$\dot{\boldsymbol{p}} = -\eta \boldsymbol{p} + g_p \text{satproj}_r^+(\boldsymbol{p}, \boldsymbol{\phi}) \tag{5.5}$$

or more explicitly:

$$\dot{\boldsymbol{p}} = \eta \boldsymbol{p} + g_p \left((\nu_r(\boldsymbol{p}) - 1) \left\langle f\left(\left\langle \boldsymbol{\phi} - \boldsymbol{p}, \hat{\boldsymbol{\phi}} \right\rangle \right) \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{p}} \right\rangle \hat{\boldsymbol{p}} + f\left(\left\langle \boldsymbol{\phi} - \boldsymbol{p}, \hat{\boldsymbol{\phi}} \right\rangle \right) \hat{\boldsymbol{\phi}} \right)$$

where $\hat{\boldsymbol{\phi}} = \boldsymbol{\phi} \|\boldsymbol{\phi}\|_2^{-1}$ and $\hat{\boldsymbol{p}} = \boldsymbol{p} \|\boldsymbol{p}\|_2^{-1}$.

We refer to the integrating system described in Equation 5.5 as the ITM_p network. Figure 5.3 shows the structure of the ITM_p network as a block diagram where the projection operator is implemented as two connected subnetworks performing a dot product and an element-wise multiplication. Notably, c^+ is a population of LIF neurons with biases sampled such that there is no activity for negative inputs, creating a non-negative threshold. In relation to the previous equations, c^+ takes over the role of $f(\cdot)$, filtering through selective neural sensitivity. Further, the subtraction portion of the projection, supposing $\langle \boldsymbol{\phi} - \boldsymbol{m}, \boldsymbol{\phi} \rangle = (1 - \langle \boldsymbol{\phi}, \boldsymbol{m} \rangle)$, is implemented as a function between c^+ and the Hadamard product subnetwork, $\odot$. The connection from c^+ to the $\odot$ subnetwork projects the scalar output of c^+ to a d-dimensional vector by multiplying it by $[1, 1, \ldots, 1]^\top$ such that every entry of $\boldsymbol{\phi}$ becomes scaled by the biased dot product.

Figure 5.4 shows a comparison between analytical and spiking neural implementations of the ITM_p (Equations 5.5 and 5.4, respectively). Just as the "satproj" system in Chapter 4 was an analytical characterization of the system realized by the spiking neurons implementing the "proj" system, Equation 5.5 provides a mathematical description of what the spiking neurons realize when implementing Equation 5.4. Evidently, the analytical implementation struggles to capture the desired primacy effects; however, the equivalent system in spiking neurons shows robust primacy effects. We attribute the lack of observed primacy effects in the analytical implementation to a combination of things: a lack of low-pass filtering of signals, a lack of consideration for soft normalization of signals within internal neural populations, and potential model mismatches. Low-pass filtering will undoubtedly change the shape of the mathematical simulation, providing an overall smoother result; however, we would not expect it to substantially affect model results. Soft normalization highlights potentially the largest difference between the neural and analytical models. Just as we leveraged ensembles' radii to alter their representational capacity (see Section 4.3), other ensembles within the ITM_p may be subject to saturation and soft normalization if the magnitude of their signals exceed their respective radii; generally, we assume maximum input magnitudes of 1 for the dot product and element-wise multiplication populations. A potential mismatch of expected magnitudes may cause an inadvertent primacy effect to arise from the dynamics by reducing the overall dot product similarities as $\boldsymbol{p}$ grows. Lastly, we note that there is minimal temporal decay of $\boldsymbol{p}$ in the spiking neural implementation. As was discussed in Section 4.3.2, the limited decay is a consequence of neural approximations as well as the small value of η from previous literature. We do note that there

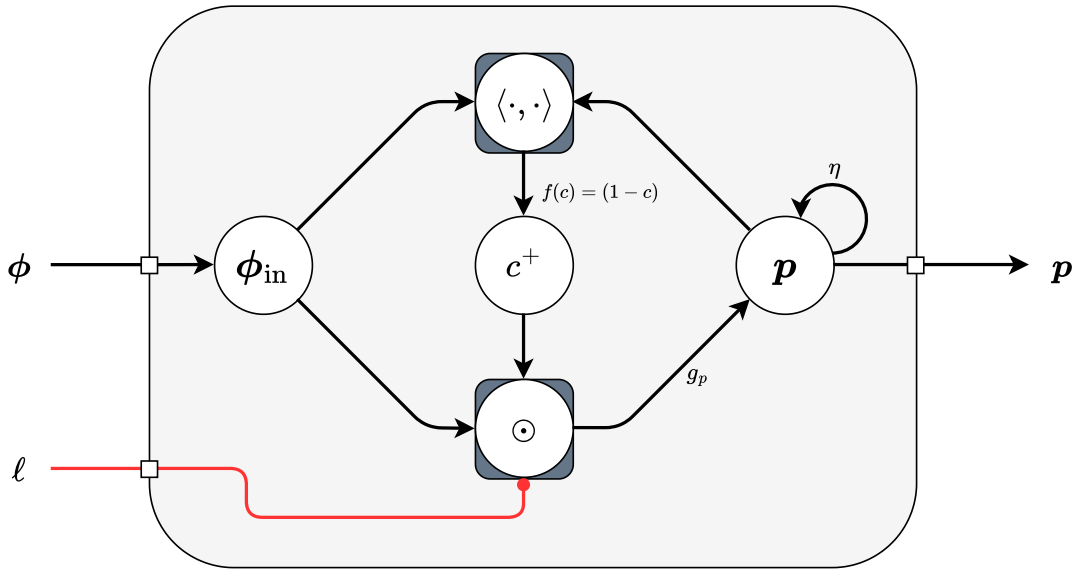


Figure 5.3: *An architectural diagram of the ITM_p network.* Input SSPs, $\boldsymbol{\phi}$, are integrated into the memory representation, $\boldsymbol{p}$, using the non-negative projection system described by Equation 5.4: $g_p f\left(\langle \boldsymbol{\phi} - \boldsymbol{p}, \boldsymbol{\phi} \rangle\right) \boldsymbol{\phi} \approx g_p \left[\left(1 - \langle \boldsymbol{\phi}_{\text{in}}, \boldsymbol{p} \rangle\right)\right]_{c^+} \boldsymbol{\phi}_{\text{in}}$ where $\boldsymbol{\phi}_{\text{in}} \approx \boldsymbol{\phi}$, $c = \langle \boldsymbol{\phi}_{\text{in}}, \boldsymbol{p} \rangle$, and $[\cdot]_{c^+}$ denotes the non-negative filtering done by the neural population c^+. When the inhibitory signal, ℓ, is active, input to $\boldsymbol{p}$ from the projection system drops to $\mathbf{0}$; however, the recurrent connection is unaffected. See Section 2.4.1 for an explanation of the symbols used within the architectural diagram.

is some significant decay from the end of item presentation ($t = 5$) to the settled value ($t \to \infty$), suggesting that there is some immediate decay and drift that negatively affects the memory.

We evaluate these potential differences by introducing low-pass filtering dynamics and soft-normalization into the analytical system's dynamics. Figure 5.5 shows what happens once we take low-pass filtering and soft-normalization partially into account by applying various low-pass filters throughout and by performing a soft-normalization on the difference between the input and the memory representation ($\boldsymbol{\phi} - \boldsymbol{p} \to \left(2 - \nu_{1,\sigma}\left(\boldsymbol{\phi} - \boldsymbol{p}\right)\right)^{-1}\left(\boldsymbol{\phi} - \boldsymbol{p}\right)$). By introducing the low-pass filtering and soft-normalization, we observe that our analytical models becomes significantly more consistent with the results of the spiking neural implementation shown in Figure 5.4 (b). Critically, these results support that the omission of soft-normalization and low-pass filtering in the analytical system formation is significantly

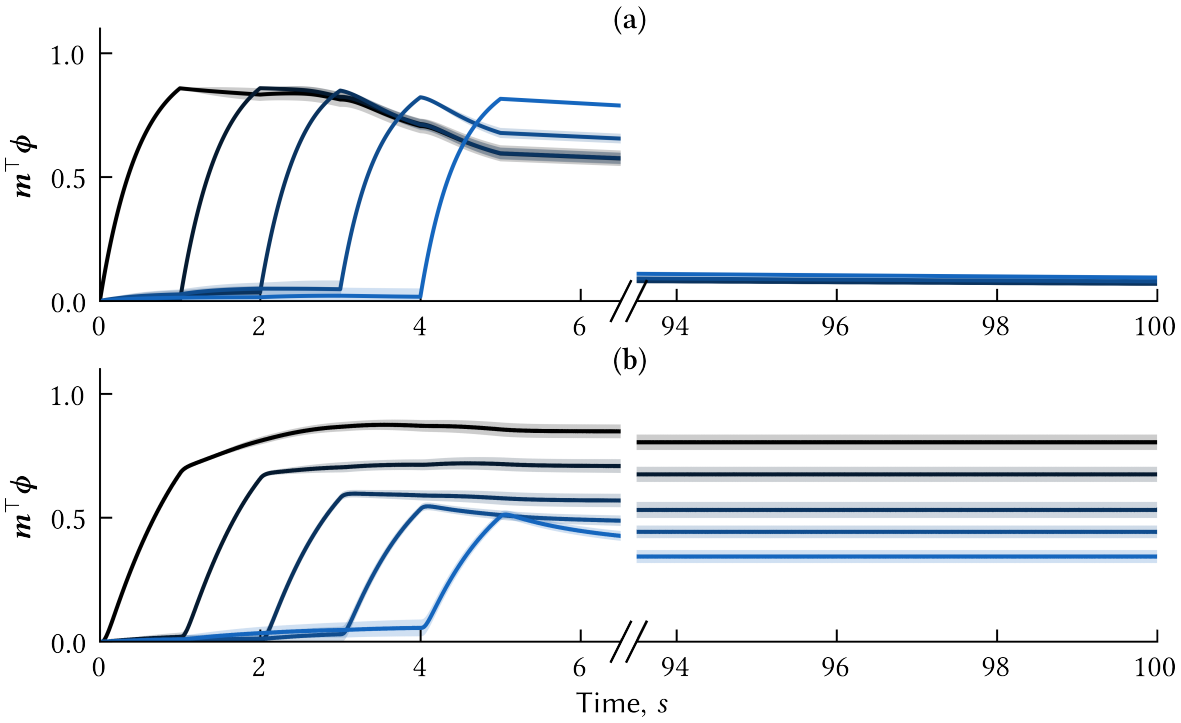


Figure 5.4: *A comparison of the analytical and spiking neural implementations for the ITM_p.* **(a)**: Dot product similarities of input vectors and the resultant memory representation for the system defined in Equation 5.5 ($d = 128$, $\eta = -2.27 \times 10^{-2}$, $g_p = 2$, $\nu_{\sqrt{2},\sigma}(\boldsymbol{p})$, and $k = 50$). **(b)**: Dot product similarities of input vectors and the neural memory representation for the system defined in Equation 5.4 implemented in spiking neurons ($d = 128$, $\eta = -2.27 \times 10^{-2}$, $g_p = 2$, $r = \sqrt{2}$, $\tau = 0.1$, and 150 neurons per dimension). Shaded regions shows 95% confidence intervals ($N = 50$).

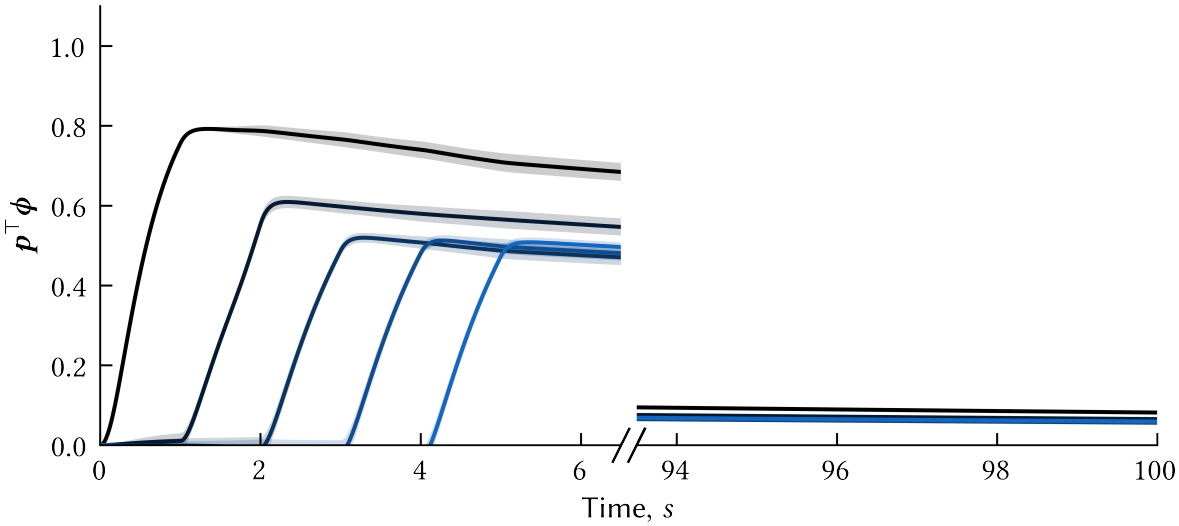


Figure 5.5: *A demonstration of the revised analytical ITM$_p$ with an evident primacy bias.* The resultant traces show dot product similarities of input vectors and the memory representation for the first and last 6 seconds of the simulation averaged over 50 randomly initialized trials. The simulated system implements Equation 5.5 with additional considerations for low-pass filtering and soft-normalization ($\boldsymbol{\phi} - \boldsymbol{p} \rightarrow (2 - \nu_{r,\sigma}(\boldsymbol{\phi} - \boldsymbol{p}))^{-1}(\boldsymbol{\phi} - \boldsymbol{p})$; $d = 128$, $\eta = -2.27 \times 10^{-2}$, $g_p = 2$, $\nu_{\sqrt{2},\sigma}(\boldsymbol{p})$, and $k = 50$). Shaded regions shows 95% confidence intervals ($N = 50$).

impactful to the analytical system's behaviour. By accounting for the low-pass filtering and the soft-normalization, we approach a system that is more consistent with the spiking neural implementation.

We note then that the primacy effect is in-part influenced by the representational capacities of the populations within the model, particularly those interacting with the memory representation populations (e.g., populations performing multiplication within the $\langle \cdot, \cdot \rangle$ subnetwork). Further, we note that this effect arises only in the ITM$_p$ and not the STM since $g_p \ll g$. That is to say, the increased input gain of the STM ($g = 50$, in previous examples) counteracts the consequences of the soft-normalization processes within the projection model.

Figure 5.6 compares neural implementations of the ITM$_p$ with varying input gains, g_p, and list lengths, N. We observe that the primacy effect in the representation degrades with increased list length as well as increased g_p. For the low gain conditions ($g_p = 2$), we see

increased similarity between the memory representations and the earliest items of the list. As the number of items increases, the similarity with the earliest items decreases. In high gain conditions ($g_p = 10$), we see little-to-no primacy biasing—each item is similarly well-represented in the memory representations for the smaller set of input vectors ($N = 5$). By contrast, the memory representation shows an evident recency bias rather than any primacy bias for high gain and larger sets of input vectors ($N = 10$). Overall, the degradation of the primacy effect reflects a decreased ability to attach to—or to stay attached to—a task's initial context through rehearsal or other means. Lastly, we note that similar to previous spiking neural simulations, there is a lack of obvious temporal decay compared to the analytical simulations; however, since the timescale of our working memory tasks is not significantly more than τ_{Decay}, we leave the optimization of η for future work.

The ITM_L and ITM_p networks form our intermediate-term memory model for our complete model; the ITM_p network is used to query tensor product associations embedded within the ITM_L network's weights during recall for all tasks:

$$L\boldsymbol{p} = \left(\sum_{i=1}^{N} \left(\left(\boldsymbol{\phi}_{1,i} + \boldsymbol{\phi}_{2,i} \right) \boldsymbol{\phi}_{1,i}^{\top} + \left(\boldsymbol{\phi}_{1,i} + \boldsymbol{\phi}_{2,i} \right) \boldsymbol{\phi}_{2,i}^{\top} \right) \right) \left(\sum_{j=1}^{N} c_j \boldsymbol{\phi}_{2,j} \right)$$

and supposing orthogonal embeddings:

$$L\boldsymbol{p} \approx \sum_{j=1}^{N} c_j \left(\boldsymbol{\phi}_{1,j} + \boldsymbol{\phi}_{2,j} \right)$$

where c_j are coefficients representing the magnitude of each SSP stored in $\boldsymbol{p}$. By consequence, we see that a query of the ITM_L network by a primacy-biased $\boldsymbol{p}$ results in a primacy-biased recollection of the associated feature vectors, enabling primacy-biased recall of list items.

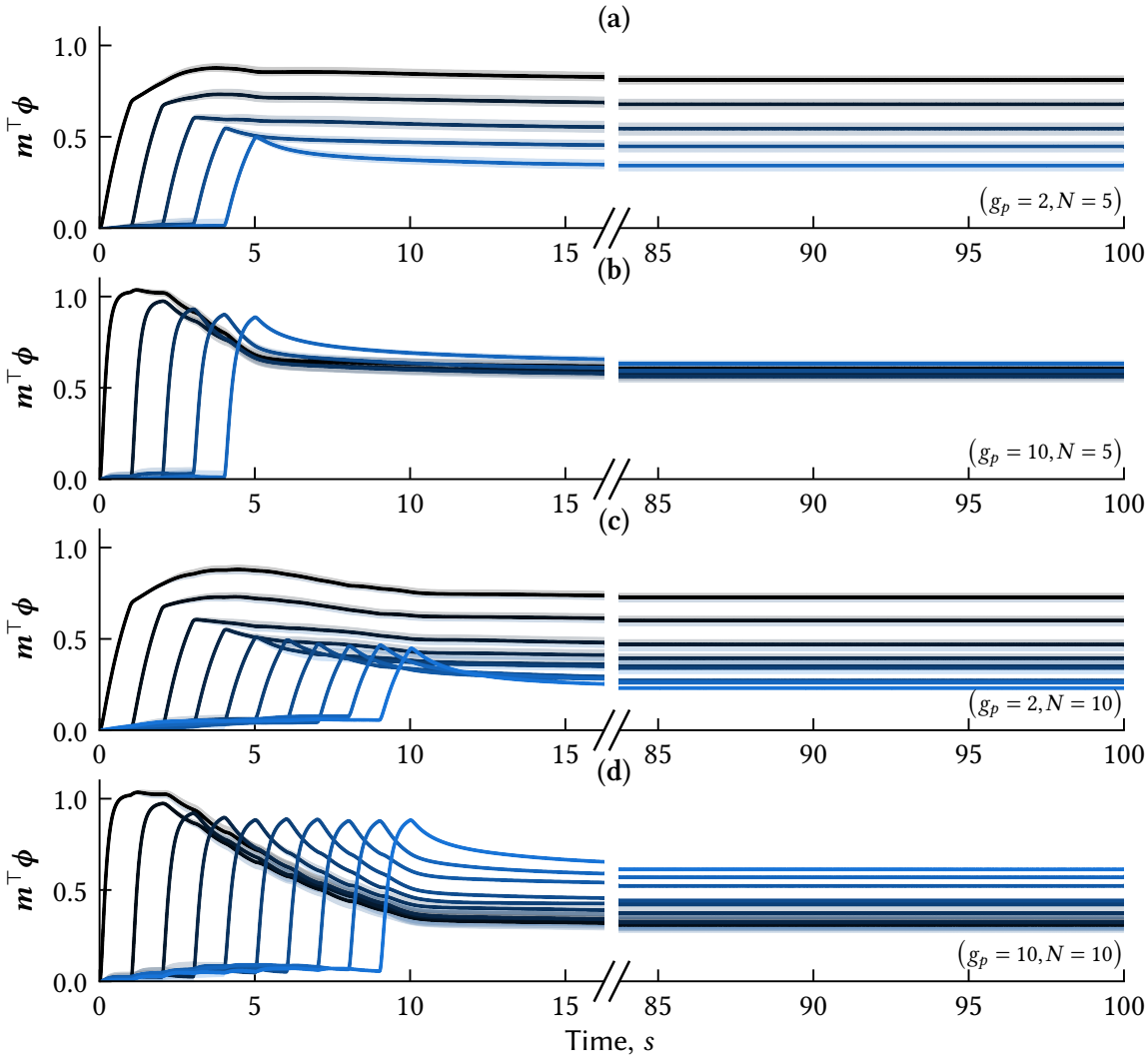


Figure 5.6: *A comparison of spiking neural ITM$_p$ behaviours for varying list lengths and gain parameters.* Lines show dot product similarities of input vectors and the resultant memory representations. All systems use $d = 128$, $\eta = -2.27 \times 10^{-2}$, $r = \sqrt{2}$, $\tau = 0.1$, and 150 neurons per dimension per population. **(a)**: Low gain system with a short list ($g_p = 2$, $N = 5$). **(b)**: Moderate gain system with a short list ($g_p = 10$, $N = 5$). **(c)**: Low gain system with a medium length list ($g_p = 2$, $N = 10$). **(d)**: Moderate gain system with a medium length list ($g_p = 10$, $N = 10$). Shaded regions show 95% confidence intervals ($N' = 50$)

Chapter 6

Semantic Memory Tasks

6.1 Methodology

To test our model on semantic memory tasks, we must first situate it within a larger network that handles input routing and recall processes. There are three external signals that control the behaviour of the model: the stimulus input signal, $\boldsymbol{\phi}_V$, the recall inhibition signal, ℓ_{R}, and the protocol inhibition signal, ℓ_{P}. The recall inhibition signal disables recall-oriented parts of the network during list presentation, and oppositely, disables input-oriented parts of the network during list recall. Alternatively, the protocol inhibition signal disables positional recall when it is not useful, i.e., when the model is attempting to recall items in serial order.

Figure 6.1 shows a simplified architectural diagram of the complete network used for semantic memory tasks, including the routing of inhibitory signals and signal querying for recall. During list presentation, SSPs are generated within the position network as the input signal changes from one list item SSP to another. Whereas during serial recall, the position network resets to the first temporal SSP and steps through temporal vectors to facilitate recall in serial order. The inner workings of the position network are further described in Gosmann and Eliasmith (2021). List item and temporal SSPs are then used as inputs to the STM and ITM_L networks. For the STM network, item vectors are bundled with the bound product of item vectors and temporal embeddings (ex. $\boldsymbol{\phi}_{\mathrm{in}} = \boldsymbol{\phi}_1 + \boldsymbol{\phi}_1 \circledast \boldsymbol{\phi}_2$ where $\boldsymbol{\phi}_1 \in \Phi_V$ and $\boldsymbol{\phi}_2 \in \Phi_T$), whereas for the ITM_L network, item vectors are just bundled with temporal embeddings (ex. $\boldsymbol{\phi}_{\mathrm{in}} = \boldsymbol{\phi}_1 + \boldsymbol{\phi}_2$). For the ITM_p network, only the temporal embeddings are used as inputs (ex. $\boldsymbol{\phi}_{\mathrm{in}} = \boldsymbol{\phi}_2$). We prevent the memory representations from becoming corrupted during recall by inhibiting populations or subnetworks within each memory model. To do so, the recall inhibition signal, ℓ_{R}, is 1 throughout the recall phase of

the simulation, whereas it is 0 during the list presentation phase. Neural populations forwarding memory output to the recall networks are inhibited throughout list presentation, as they are inversely inhibited by ℓ_{R} (i.e., they are inhibited when $\ell_{\mathrm{R}} = 0$ and not inhibited when $\ell_{\mathrm{R}} = 1$). For the serial recall task, the recall of temporal SSPs and the use of recalled list item SSPs to recall temporal SSPs are not particularly beneficial, as with the ITM_p network's querying, SSP recall would be biased towards the latest SSPs. To counteract this effect, we suppress the positional recall network as well as the feedback population by which recalled items can be used to prompt the ITM_L network.

The recall networks within our larger network implement Independent Accumulator networks (IAs), where a competitive, leaky process of evidence accumulation that cleans up an input vector to an item within a lexicon with sufficient similarity (Gosmann et al., 2017). Similar to prior work, we use two recall networks—one for list item SSPs and one for their associated temporal/positional SSPs—where the temporal recall network call network is used for recalling items in free recall task paradigms. To do so, recalled positional vectors act as query vectors and prompt our memory networks for their related information. The output of the list item recall network is recorded to determine which items were recalled as well as to determine the order of recall.

To capture human-like limitations, we impose no structural changes and limited parameter changes on the model between serial and free recall tasks. Initial parameter values were selected to achieve the individual memory model components' intended behaviours—e.g., loading an entire SSP into the projection memory network or learning an ITM_L network mapping within 300ms—and were incrementally tuned across tasks based on the model's results. Differences in parameter values between tasks reflect differences in experimental paradigms: an increase to the ITM_p network's gain parameter in free recall due to rehearsal interruption; and an increase to the minimum evidence of the recall networks for serial recall due to an increased desire for positional accuracy. Table 6.1 provides a summary of the model and task parameters for the semantic memory tasks.

6.2 Serial List Recall

For the serial recall task, we examine the average number of correctly recalled items per trial, the proportion of human and model task results that recalled items in the correct serial position—the serial position curve, and the proportion of total item responses that were recalled in the correct position—the positional accuracy. We use traditional, Gaussian 95% confidence intervals (CIs) for comparing the number of recalled items per trial; however, the remaining points of comparison are proportions—means of binomial vari-

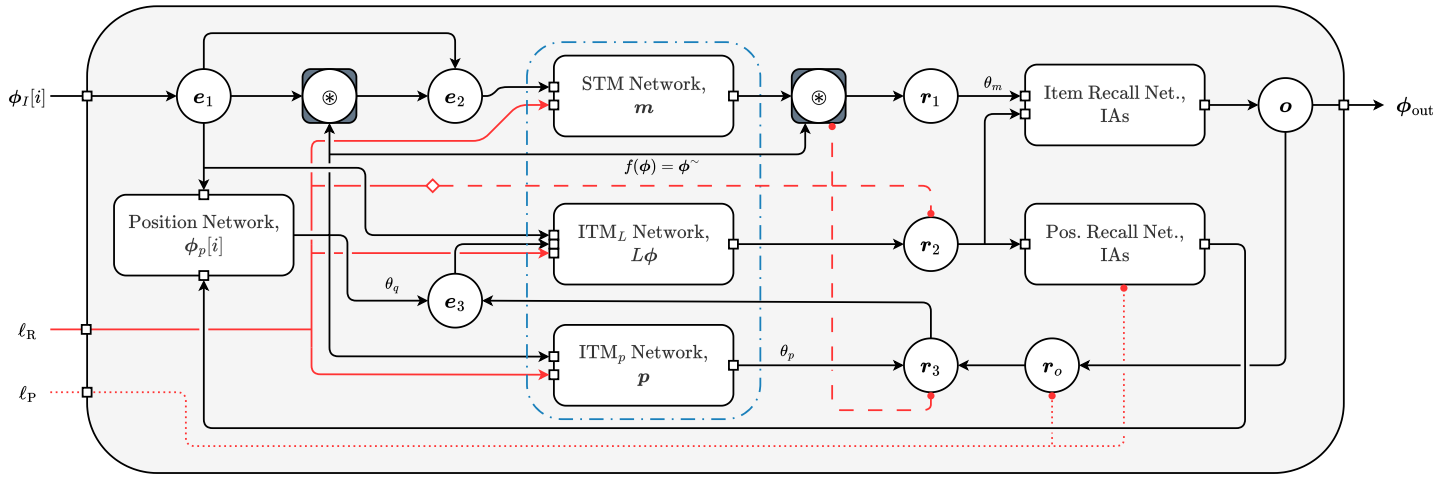


Figure 6.1: *An architectural diagram of the integrated models for semantic memory tasks.* Several populations and subnetworks facilitate input processing and information routing for different types of list recall and different phases of the test. The blue dot-dash box highlights the memory networks developed in Chapters 4 and 5.

Table 6.1: *Model Parameters for Semantic Memory Tasks*

Parameter	Serial	Free	Description	Reason for Change
d	256	256	(S)SP Dimensionality	—
l	10	10	AML Learning Rate, ITM_L	—
α	1	1	AML Learning Threshold, ITM_L	—
η	-0.0228	-0.0228	Natural Decay Parameter, STM & ITM_p	—
g_m	50	50	Input Gain, STM	—
g_p	2	10	Input Gain, ITM_p	Rehearsal Interruption
$N_{m,p}$	4	4	Capacity (chunks), STM & ITM_p	—
θ_m	1.0	1.0	Contribution to Recall, STM	—
θ_p	0.707	0.707	Contribution to Recall, ITM_p	—
θ_q	1.0	1.0	Contribution to Recall, $\boldsymbol{\phi}_T$	—
μ_{IA}	0.375	0.30	Minimum Evidence, IAs	Serial Position Specificity
σ_{IA}	0.009	0.009	Recall Noise, IAs	—

ables. We avoid assuming that the mean is normally distributed, as this could lead to CIs including negative values or values above 1. Instead, we calculate 95% CIs for proportion metrics using the Clopper-Pearson Exact Method (Clopper & Pearson, 1934). We use these Clopper-Pearson confidence intervals (CP-CIs) to qualitatively compare serial position curves and positional accuracies. Since CP-CIs cannot be trivially mixed to evaluate statistically significant differences between human and model data, we calculate 95% CIs for differences in proportions using the Newcombe-Wilson method—a less conservative means of hypothesis testing (See Method 10 of Newcombe, 1998). If a Newcombe-Wilson confidence interval (NW-CI) does not contain 0, we can conclude that there is a statistically significant difference between individual data points. Since significance tests are not meaningful in isolation, we calculate Cohen's h to assess the effect size of differences in proportion metrics—significant or not (Cohen, 2013). Since the average number of correctly recalled items can also be understood as a proportion of the list length, we calculate Cohen's h to evaluate its respective effect size. Based on Cohen's suggested interpretations—$|h_S| = 0.20$ is a small effect size; $|h_M| = 0.50$ is a moderate effect size; and $|h_L| = 0.80$ is a large effect size—we average neighbouring $|h|$ values to determine the limits of what we consider small, moderate, and large effect sizes. We group small effect sizes as $|h| < 0.35$, moderate effect sizes as $0.35 \leq |h| < 0.65$, and large effect sizes as $0.65 \leq |h|$. In doing so, we establish thresholds between the different effect sizes consistent with their original definitions.

Following the protocol of Jahnke (1968), we present the complete model with a list of 10 items at a rate of 1 item per second. Following this presentation, the model attempts to

recall list items in serial order. We then assess its responses based on the output of the list item recall network. We ran our model for 100 different random initializations. Because neurons' parameters and SSPs' embeddings are formed probabilistically, we expect some variation in model performance between trials. By running for many random initializations, we approach a better overall view of the model's performance and are more capable of comparing it to the results of Jahnke (1968), for which there were 96 recorded results. We gather the model's results for individual tasks, and we present our analysis of these results in the following section.

6.2.1 Results

Figure 6.2 provides a visual comparison of the serial position curves between human and model responses as well as the distribution of transposition errors. The serial position curve shows the proportion of trials in which an item was recalled as a function of its list position, p_{Si} for $i \in \mathbb{N}_{\leq 10}$, whereas the distribution of transposition errors shows the proportion of recalled items that were recalled in positions relative to their true position, p_{Tj} for $j \in \mathbb{Z}$ where $-6 \leq j \leq 6$. Qualitatively, the model's serial position curve demonstrates the distinctive $\cup$-shaped curve that is characteristic of human serial position curves. The model demonstrates slightly weaker than, although not significantly different from, human performance. The model recalls, on average, 3.55 ± 0.32 list items compared to the reported human average of 3.83 ± 0.35 (95% CIs, $|h| = 0.06$). Also, the serial position curves show considerable CI overlap for the experimental and model results, suggesting limited significant differences between them. While Jahnke (1968) does not report specific transposition error data, they provide their rate of positional accuracy, and the resultant CP-CI is $(0.78, 0.86)$. The model shows a similar rate of positional accuracy with a CP-CI of $(0.82, 0.90)$. These ranges show significant overlap, and similar to the previously explored results, suggest strong similarity between model and experimental results for the serial recall task. We additionally note that our transposition error distribution is consistent with various other serial recall experiments in which transpositions were unlikely, localized, and slightly biased in the forward direction (Farrell & Lewandowsky, 2004; Henson et al., 1996).

Figure 6.3 shows the statistical analysis of the serial recall results. Despite CP-CI overlap for all points on the serial position curve, we see statistically significant differences between two of the serial position curve points when calculating NW-CIs, namely the proportions of trials in which the 2^{nd} and 9^{th} list items are recalled, p_{S2} and p_{S9} respectively. Cohen's h values show small effect sizes for all metrics except p_{S9}: $0.03 < |h_{T_0}|, |h_{S\neq 9}| < 0.31$. For p_{S9}, the Cohen's h suggests a moderate effect size: $|h_{S9}| = 0.41$. See Table A.1 in Appendix A for a complete report of confidence intervals and effect sizes. Across the serial

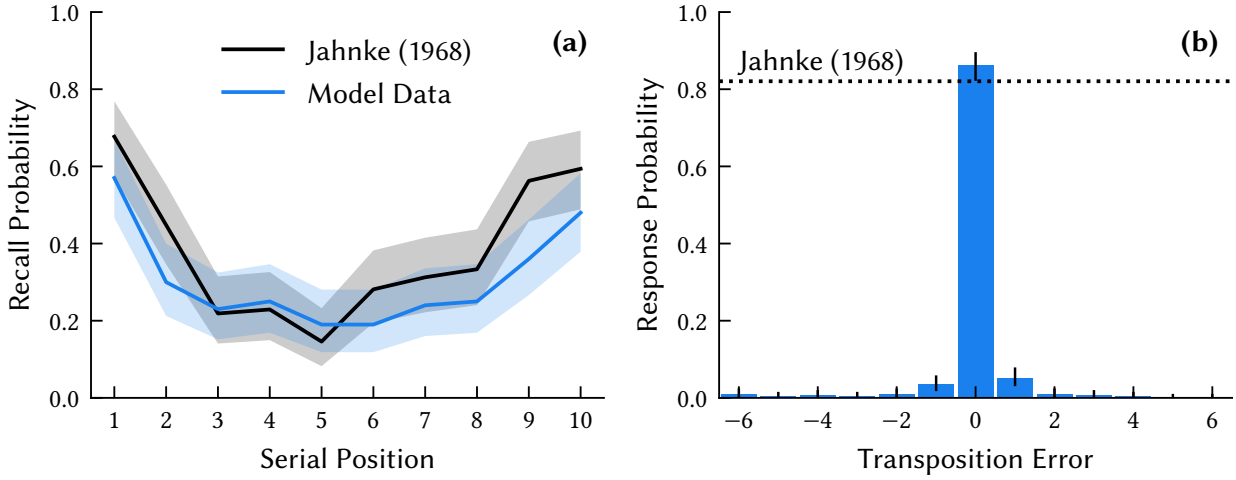


Figure 6.2: *Model and experimental serial recall results.* **(a)**: Serial position curves for model and experimental results. Lines show proportions of responses that correctly recalled a given item in its correct serial position. Shaded regions show 95% CIs using the Clopper-Pearson Exact method (Model: $n = 100$, Jahnke: $n = 96$). **(b)**: Transposition error distribution for model results. Bars show proportion of responses recalled as a function of their placement during recall. The dashed line shows the positional accuracy of responses in experimental results. Error bars show 95% CIs using the Clopper-Pearson Exact method (Model: $n = 355$, Jahnke: $n = 368$).

recall results, 2 of 12 metrics demonstrate statistically significant differences; however, only 1 of 12 metrics demonstrate a non-small effect size. For all 12 data points, we find a mean and median absolute Cohen's h of 0.17 and 0.17 respectively. Overall, these results support our model's ability to exhibit human-like serial recall abilities.

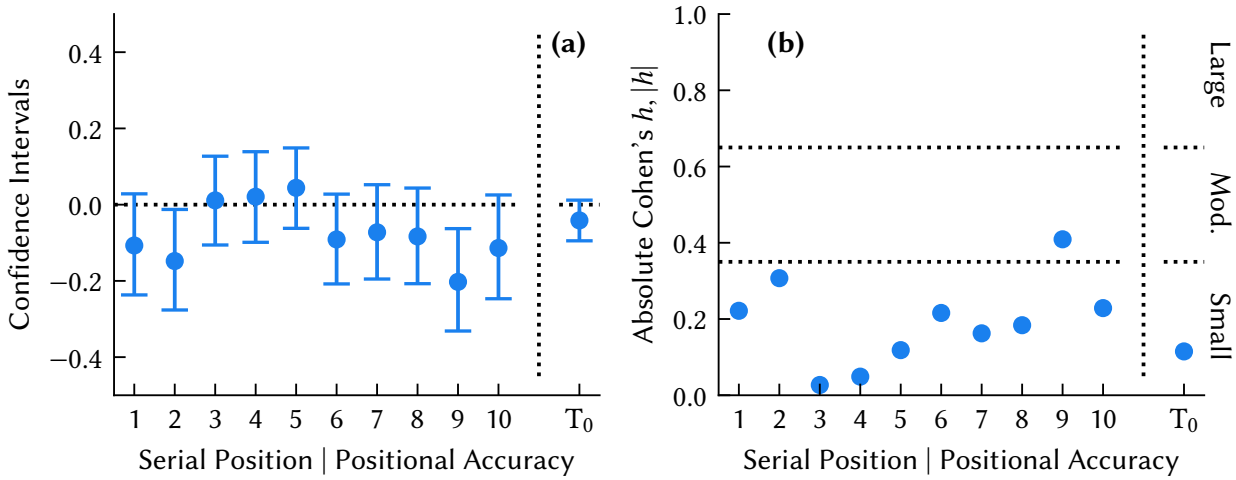


Figure 6.3: *Confidence intervals and absolute Cohen's h for proportion differences between model and experimental serial recall results.* **(a)**: Newcombe-Wilson 95% confidence intervals for differences in serial position curve and positional accuracy metrics. **(b)**: Absolute Cohen's h values for proportion estimates between serial position curve and positional accuracy metrics.

Since Choo (2010) and Gosmann and Eliasmith (2021) both use Jahnke (1968) as a benchmark for comparison as well, we additionally compare our results to their published results in order to get a more complete picture. Figure 6.4 shows the serial position curve for Jahnke (1968) compared against the three models: the STM+ITM (Our model), the snOSE (Choo, 2010), and the CUE (Gosmann & Eliasmith, 2021). All three models demonstrate evident primacy biases in recall, while only our model and the CUE model demonstrate an evident recency bias. Both our model and the CUE model fit well to the experimental data; however, the CUE model does have a slightly stronger recency bias, making it a stronger overall fit to the experimental data. Calculating the average absolute Cohen's h for just the serial position curve across the three models, we find: $|h_{\text{STM+ITM}}| \approx 0.19$, $|h_{\text{OSE}}| \approx 0.42$, and $|h_{\text{CUE}}| \approx 0.16$. These average effect sizes support that both our model and the CUE model well-reflect the experimental data; however, the CUE model is slightly more consistent with the average participant from the original experiment.

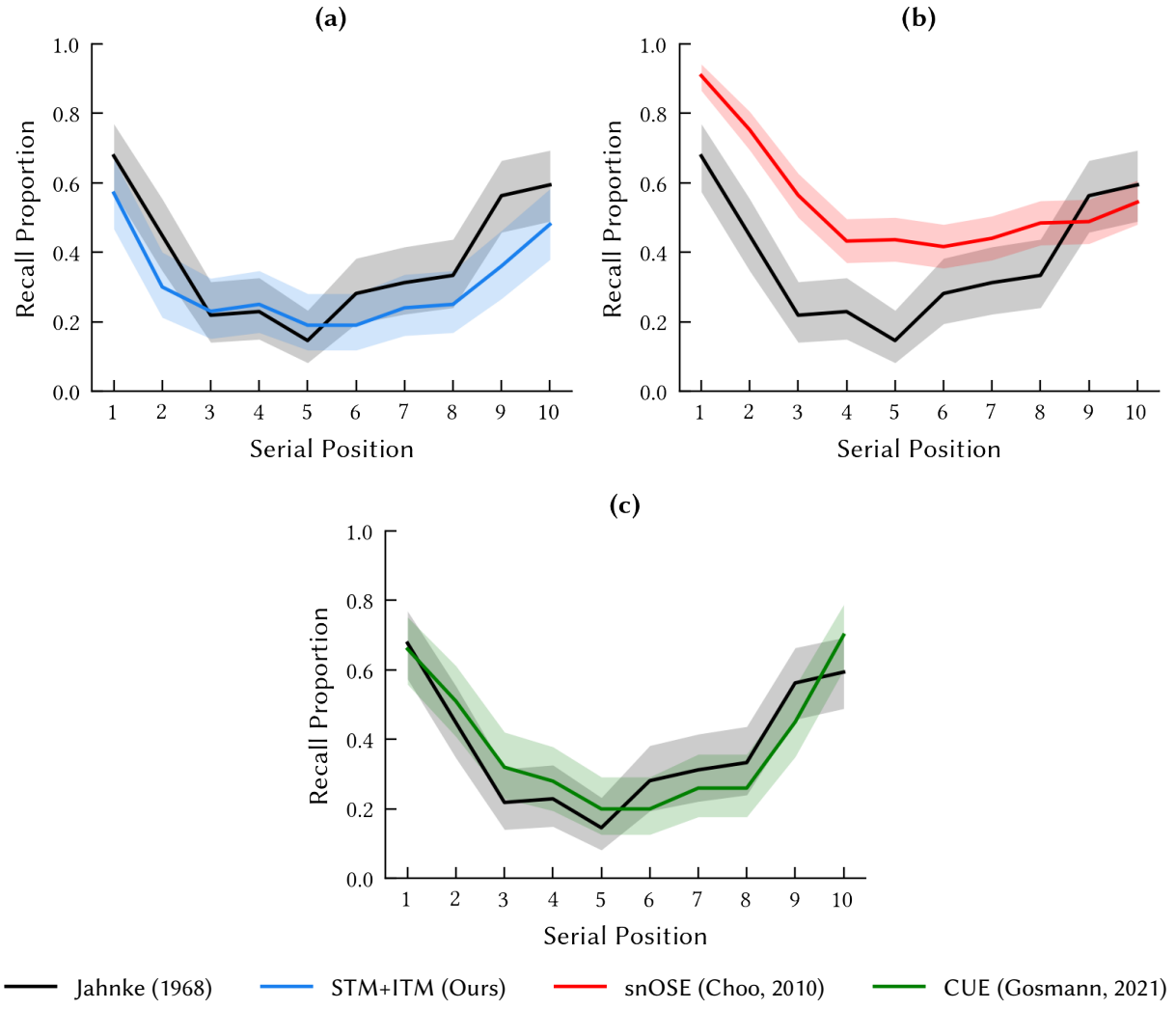


Figure 6.4: *A comparison of STM+ITM, snOSE, and CUE models to experimental serial recall results.* **(a)**: Serial position curves comparing STM+ITM model (Ours) and experimental results (Jahnke, 1968). **(b)**: Serial position curves comparing snOSE (Choo, 2010) and experimental results **(c)**: Serial position curves comparing CUE (Gosmann & Eliasmith, 2021) and experimental results. Shaded regions show 95% CIs using the Clopper-Pearson Exact method (STM+ITM: $n = 100$, snOSE: $n = 250$, CUE: $n = 100$, Jahnke: $n = 96$).

.

6.3 Free List Recall

For the free recall task, we examine the average number of recalled list items per trial; the proportion of trials that recalled a given item with respect to its list position (serial position curve); the proportion of trials that recalled a given item first with respect to its list position (first recall curve); and the proportion of individual responses that had a given position relative to the previously recalled item's position (conditional response curve). Just as we calculated CP-CIs for individual data points, NW-CIs for differences in data points, and Cohen's h corresponding to each difference for the serial recall task results, we repeat the analysis process for the free recall task results.

Following the protocol of Experiment 1 from Howard and Kahana (1999), we present the model with a list of 12 items at a rate of 1 item per second. Following this presentation, the model attempts to recall them in any order, relying on the ITM_p and recalled temporal vectors. As described in the previous section, we assess its responses based on the output of the list item recall network. Consistent with the serial recall task, we ran the task for 100 different model initializations. While this is significantly fewer than the approximately 780 trials recorded by Howard and Kahana (1999), our use of effect sizes to compare proportions is independent of sample sizes. Further, we do not assume equal sample sizes when calculating CP-CIs or NW-CIs. If we were to run an increased number of trials, we may expect CIs to shrink around, or near, the repeated proportions for the 100 original trials. Consequently, we may see an increased number of statistically significant differences between datums, but we would not expect effect sizes to change significantly. We present the analysis of our results in the following section.

6.3.1 Results

Figure 6.5 provides a visual comparison of the serial position curves, the first recall curves, and the conditional response curves between human and model responses. As was the case in the serial list recall experiments, the serial position curve shows the proportion of trials in which an item was recalled as a function of its list position, p_{Si} for $i \in \mathbb{N}_{\leq 12}$; the first recall curves show the proportion of trials in which a list item was recalled first as a function of its position in the list, p_{Fj} for $j \in \mathbb{N}_{\leq 12}$; and the conditional response curve shows the proportion of responses that were recalled as a function of their position relative to the previously recalled item, p_{Ck} for $k \in \mathbb{Z}$ where $-6 \leq k \leq 6$. Firstly, the serial position curve of the model has a shape that is qualitatively consistent with the human serial position curve—namely a distinctly reduced primacy effect despite a maintained recency effect. Secondly, the first recall curves show a large tendency of the model to recall

the last list items first—as well as a small tendency of the model to recall the first item first. Thirdly, the model demonstrates a forward-biased conditional response proportion; however, the model results are noisier and more disjointed than the human results. Lastly, the model demonstrates an average number of recalled items, 4.49 ± 0.32, which is highly similar to the average number of human recalled items, 4.51 ± 0.11 (95% CIs, $|h| = 0.003$). The CP-CIs show considerable overlap for 30 of the 36 analysed data points. For the serial position curves, the 11^{th} and 12^{th} entries are non-overlapping; for the first recall curves, all items show overlap; and for the conditional response curves, the -1^{st}, 1^{st}, 4^{th}, and 6^{th} points are non-overlapping. Despite the non-overlap for these points of comparison, our model is largely consistent with the experimental data. Our model demonstrates a degraded primacy bias; an intact recency bias; higher probability of recalling terminal items first; and CRP asymmetry typical of human behaviour in free recall tasks (Golomb et al., 2008; Howard & Kahana, 1999).

Figure 6.6 shows the NW-CIs for the proportion differences of the free recall results. Critically, we see 11 instances of statistically significant differences across the serial position curves (3), the first recall curves (3), and the conditional response curves (5). Despite these significant differences, we note that many of these NW-CIs are narrow with an extent near zero. As an example, the difference CIs for all the significantly different points of the first recall curves have an extent within 0.005 from 0, suggesting that despite the statistically significant difference in these data points, the difference is minimal.

Figure 6.7 shows the absolute Cohen's h values for the three free recall curves. Across the curves, we find that only three data points demonstrate non-small effect sizes: $p_{S11} = 0.49$, $p_{S12} = 0.64$, and $p_{C6} = 0.43$. The remainder of absolute Cohen's h measures fall into the range $0.002 < |h| < 0.31$. For all 36 data points, we find a mean and median absolute Cohen's h of 0.15 and 0.11 respectively. See Table A.2 in Appendix A for a complete report of confidence intervals and effect sizes. Consistent with serial recall task results, the distribution of effect sizes suggests a strong alignment between model and experimental results. Overall, these results support our model's ability to exhibit human-like performance in free recall.

Lastly, we compare to the results of Gosmann and Eliasmith (2021) for the same experimental paradigm. Figure 6.8 shows the serial position curves, the first recall curves, and the conditional response curves for our model (STM+ITM) and the CUE model compared to the human experimental results. Both models demonstrate evident recency effects, degraded primacy effects, a preference for recalling the most recent or first items of the list first, and a local, forward bias in recalling successive list items. Consistent with the serial recall results, both models fit well to the experimental data with the CUE model performing slightly better on serial position and conditional response curves. Calculating the average

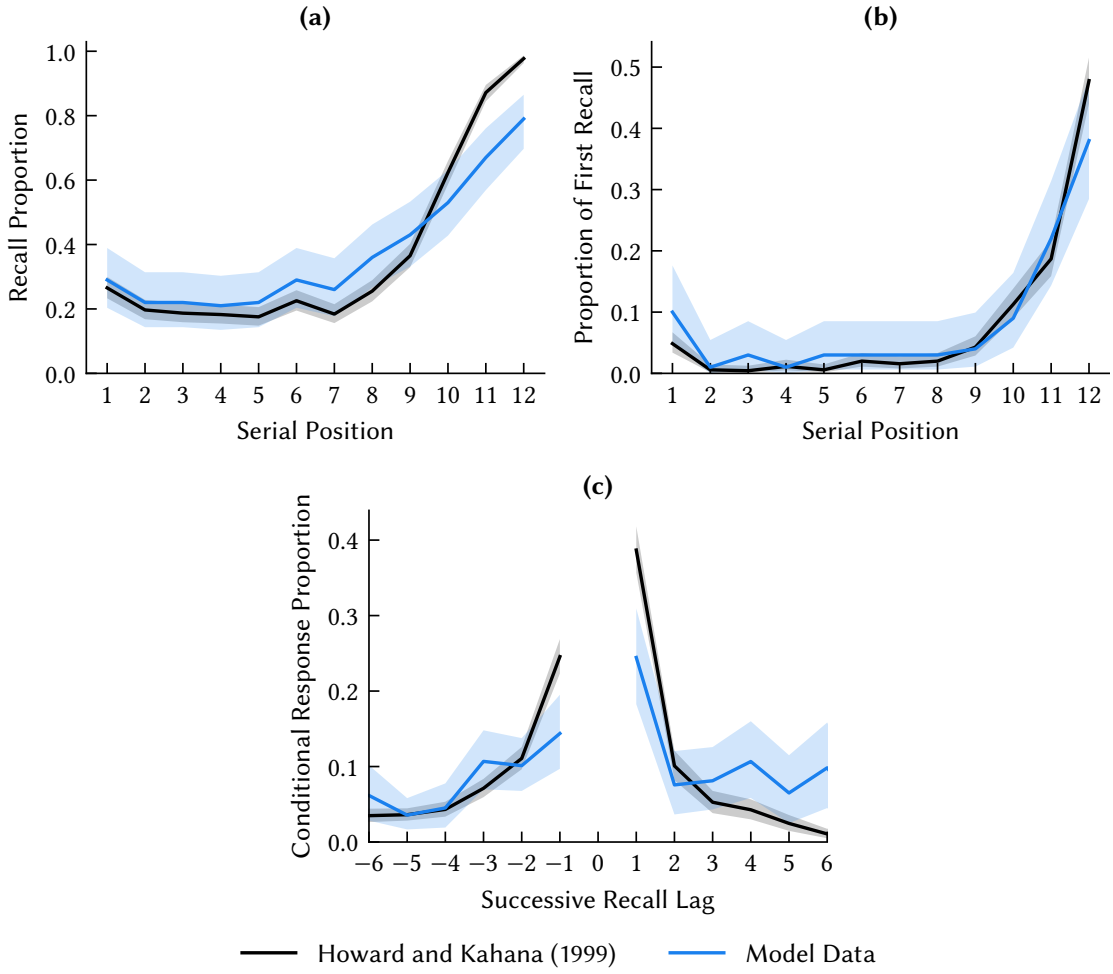


Figure 6.5: *Model and experimental free recall results.* **(a)**: Serial position curves for model and experimental results. **(b)**: First recall curves for model and experimental results. **(c)**: Conditional response curves for model and experimental results. Shaded regions show 95% CIs using the Clopper-Pearson Exact method (Model: $n = 100$, Howard & Kahana: $n = 701$).

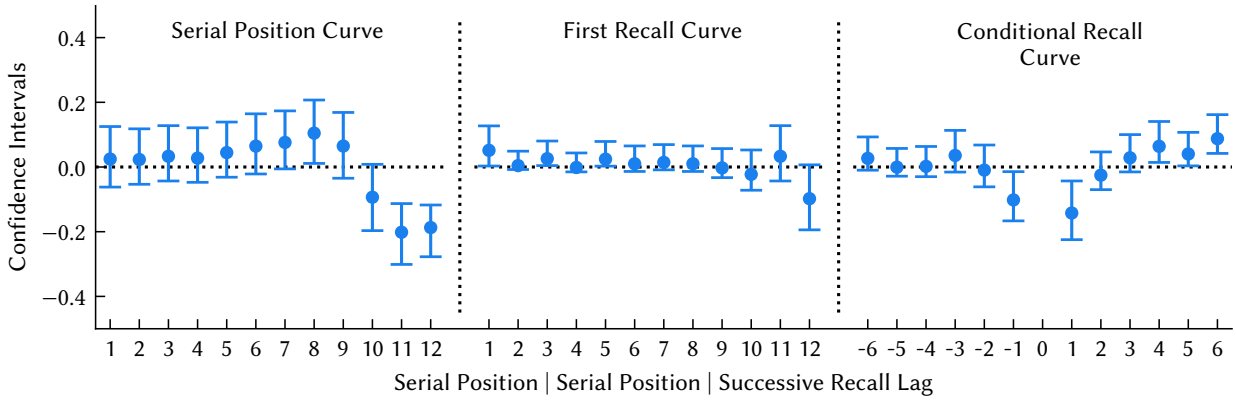


Figure 6.6: *Confidence intervals for proportion differences between model and experimental free recall results.* Newcombe-Wilson 95% confidence intervals for differences in serial position curves, first recall curves, and conditional response curves.

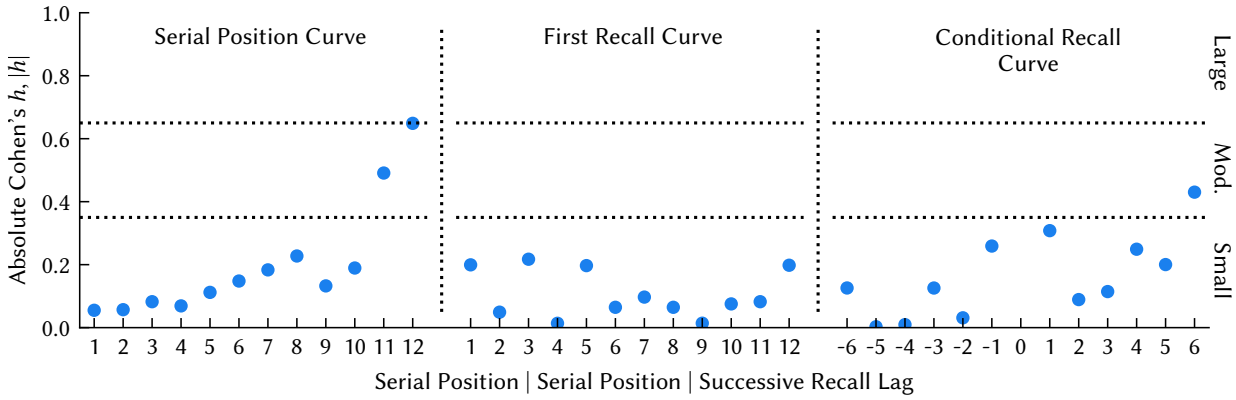


Figure 6.7: *Absolute Cohen's h for proportion differences between model and experimental free recall results.* Cohen's h for differences in serial position curves, first recall curves, and conditional response curves.

absolute Cohen's h for the three curves, we find $|h_{\text{STM+ITM}}| \approx 0.16$ and $|h_{\text{CUE}}| \approx 0.11$. These average effect sizes support that both our model and the CUE model well-reflect the experimental data, but we note that similar to the serial recall results, the CUE model is slightly more consistent with the average participant from the original experiment.

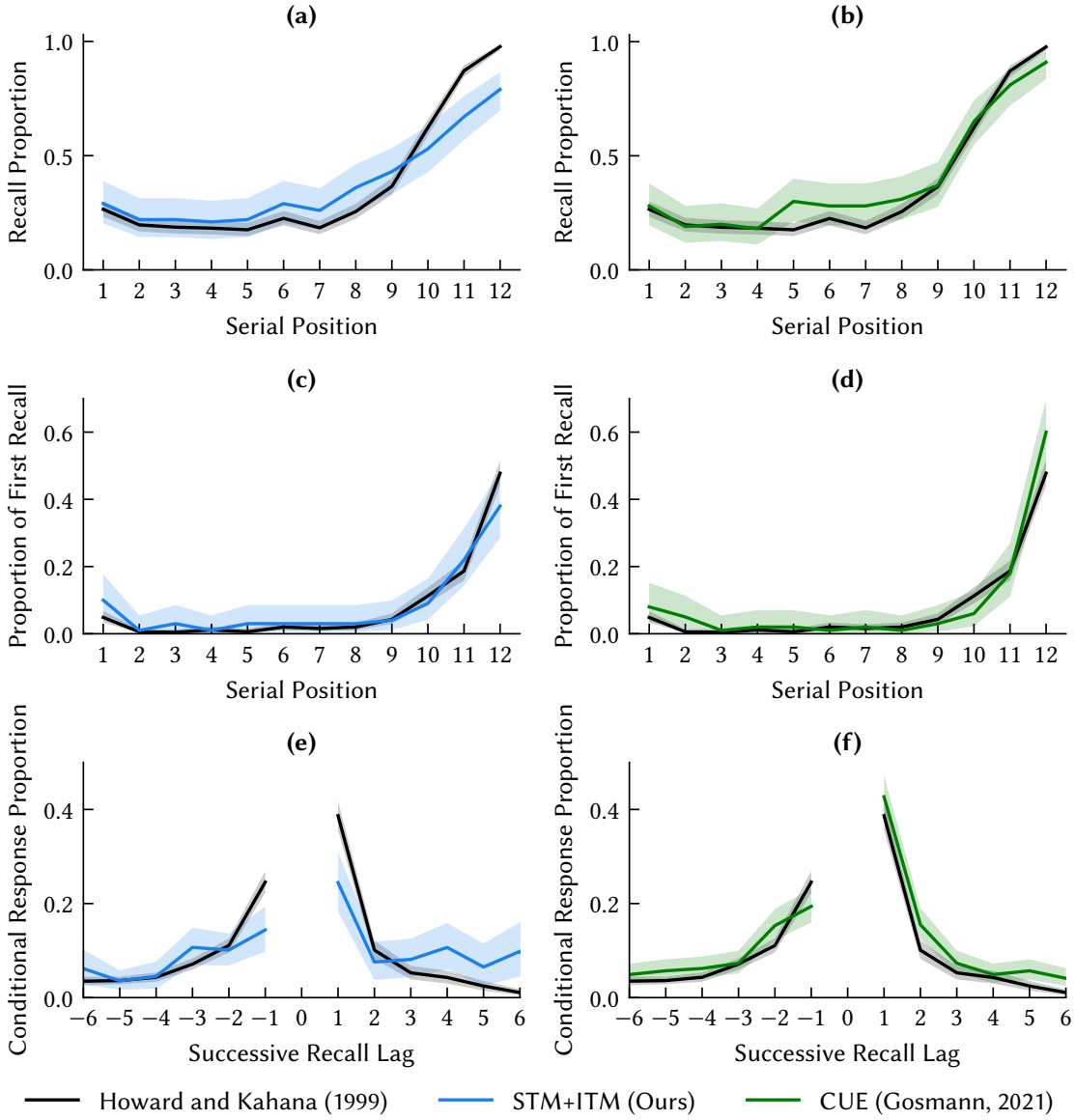


Figure 6.8: *A comparison of STM+ITM and CUE models to experimental free recall results.* **(a)** and **(b)**: Serial position curves comparing experimental, STM+ITM model (Ours), and CUE model results (Gosmann & Eliasmith, 2021). **(c)** and **(d)**: First recall curves comparing experimental, STM+ITM model (Ours), and CUE model results. **(e)** and **(f)**: Conditional response curves experimental, STM+ITM model, and CUE model results. Shaded regions show 95% CIs using the Clopper-Pearson Exact method (STM+ITM: $n = 100$, CUE: $n = 100$, Howard & Kahana: $n = 701$).

Chapter 7

Spatial Memory Task

7.1 Methodology

To test our model on a spatial memory task, we must adapt the complete model presented in Chapter 6 to integrate it within the snSLAM framework (Dumont et al., 2023). The snSLAM is a spiking SLAM model that takes in a velocity signal, estimates its self-position, learns the positions of items in the environment, and corrects its accumulated self-position error upon seeing an item again. The snSLAM is composed of several components serving different purposes: Localization, Landmark Perception, Environment Mapping, and Loop Closure. We leave Localization and Loop Closure components largely unchanged, as they provide an estimate of self-position from the velocity signal and calculate the difference between the estimated self-position and the self-position estimatable from a recalled landmark's position. We alter the landmark perception component, introducing the binding necessary for the STM network as well as the connections necessary to pipe the signals into each of our model's components. We fully replace the existing associative memory that learned the environment map with our model. It is worth highlighting that the environment map will be embedded into the weights of the auto-associative ITM_L network just as landmark-to-position associations were embedded into the weights of the previous memory.

The complete spatial task architecture is structurally very similar to the semantic task architecture except for: the loop closure component correcting self-position when encountering a remembered landmark, and the ego/allocentric representational distinction between perceptual and memory systems. Figure 7.1 shows a simplified architectural diagram of the complete network used for the spatial memory task. The model itself receives five external signals: the landmark input signal, $\boldsymbol{\phi}_L$; the egocentric landmark position signal,

$\boldsymbol{\phi}_{\hat{S}}$; the velocity signal, $\boldsymbol{v}$; the landmark visibility inhibition signal, ℓ_{L}; and the recall inhibition signal, ℓ_{R}. Based on the velocity signal, the position network updates a set of velocity-controlled oscillators implementing a path integrator, creating an estimate of the agent's position in the space as an SSP. Since the path integrator will drift slightly, the snSLAM model uses a grid cell-based clean-up network that removes noise. As the agent navigates between landmarks, the landmark visibility inhibition signal is active, as there are no visible landmarks to remember until the agent reaches a landmark. Once the agent reaches a landmark, it loads the bound product of the landmark and its allocentric position as an SSP into the STM network as well as their bundle into the ITM_L network. The ITM_p network, meanwhile, integrates spatial locations of the visited landmarks.

As per the augmented reality protocol of Maidenbaum et al. (2025), our modified snSLAM agent performed a spatial memory task in which it was initialized in the center of a 7×11m room and followed some provided 2D velocity profile. We generated the velocity profile based on the average walking speed of an adult and the uniform random placement of four distinct landmarks. We then convolved the velocity profile with a Gaussian kernel to smooth the velocity profiles. In each trial, the agent visited every landmark once, each of which were only visible for the 1500ms that the agent spent visiting them. As an analogue to the distractor task of Maidenbaum et al. (2025) in which participants followed a moving rabbit in the augmented reality environment, our agent followed a moving landmark in a random direction for a minimum of 5m. Once this distractor task was complete, we queried the agent for its estimate of each landmark's position relative to its own position in order of their visitation. Since we have knowledge of its ground truth position, we can then calculate its response in the allocentric space and evaluate its response accuracy.

Due to the structural, representational, and methodological differences between the semantic and spatial memory tasks—namely the shift from random SSPs to Hexagonal SSPs and the presence of a distractor task—we revisited four model parameters. The transition from representing a 1D space using random SSPs to a 2D space motivated both the use of Hexagonal SSPs and an increase to the number of dimensions (338) compared to the previously used random SSPs (256). These changes allowed us to achieve smooth kernel shapes over the 2D space. Because we increase the number of dimensions, we must also revisit the AML learning rate of the ITM_L network since it depends on the magnitude of individual vector entries. Since the distractor task degrades the learned ITM_L network associations and compromises the activities of the STM and ITM_p networks, we decrease the contributions of the STM and ITM_p networks to recall. Just as we tuned model parameters for the semantic recall tasks, we adjusted initial estimates based on model results until we achieved human-like performance. Table 7.1 summarizes the parameters used for the spatial recall task.

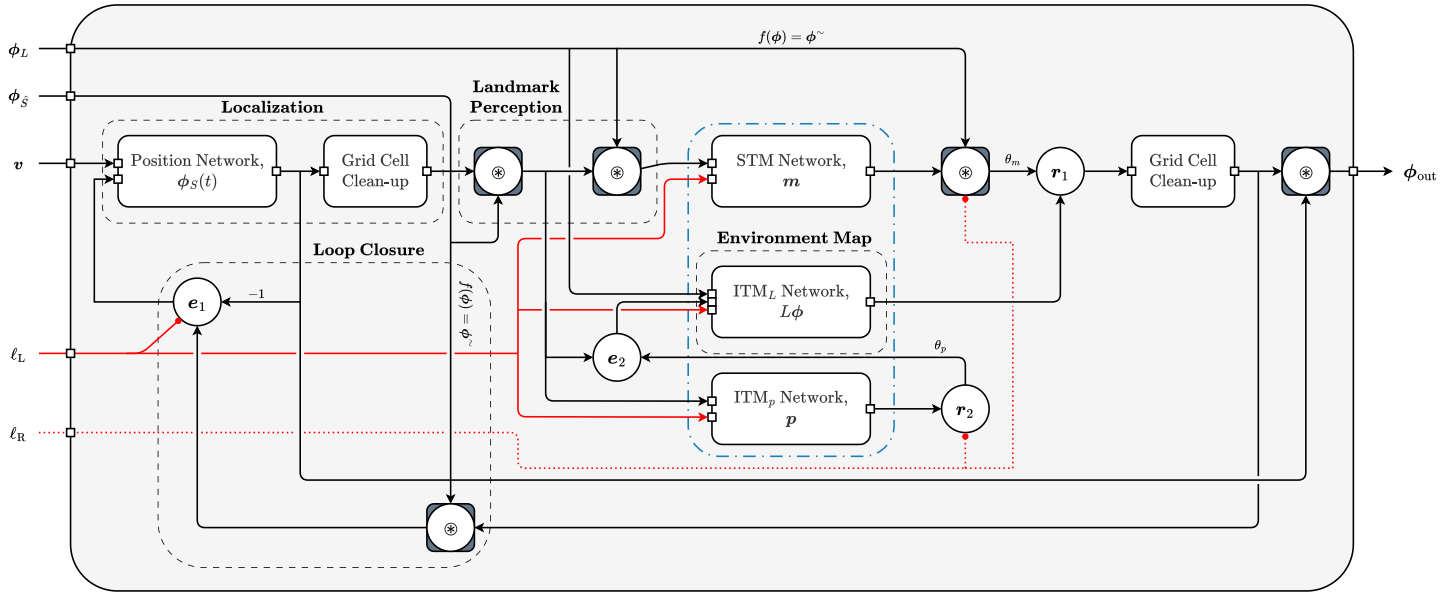


Figure 7.1: *An architectural diagram of the integrated models for spatial memory tasks.* Several populations and subnetworks facilitate input processing and information routing for remembering and recalling observed landmarks and their associated positions. The black dashed boxes highlight portions of the model corresponding to their original functions (See Figure 2 of Dumont et al., 2023). The blue dot-dash box highlights the memory networks developed in Chapters 4 and 5.

Table 7.1: *Model Parameters for the Spatial Memory Task*

Parameter	Spatial	Description	Reason for Change
d	338	(S)SP Dimensionality	Hexagonal SSP Confusability
l	30	AML Learning Rate, ITM_L	Δd
α	1	AML Learning Threshold, ITM_L	—
η	-0.0228	Natural Decay Parameter, STM & ITM_p	—
g_m	50	Input Gain, STM	—
g_p	2	Input Gain, ITM_p	—
$N_{m,p}$	4	Capacity (chunks), STM & ITM_p	—
θ_m	0.1	Contribution to Recall, STM	Distractor Task
θ_p	0.1	Contribution to Recall, ITM_p	Distractor Task
θ_q	1.0	Contribution to Recall, $\boldsymbol{\phi}_T$	—

Performance in Maidenbaum et al. (2025) was assessed by calculating a Corrected Error Distance (CED) for each response of each participant. The CED is a measure of response quality by taking into account the distribution of possible responses. Since landmarks were randomly placed in the room, the use of Euclidean distance is unreliable, as the positioning of the item determines the distribution of possible distances. CED converts the raw Euclidean distance to a percentile-based estimate by subtracting the percentile of the response with respect to all possible responses from 1. That is to say, if a participant recalled a landmark being in a certain position, that position would be in the i^{th} percentile of possible responses, and the CED would be $1 - i \times 10^{-2}$. We adopt this same performance metric, and since it is equivalent to the proportion of possible responses that would have been better than a given response, we can treat the Mean CED (MCED) as a proportion metric and analyze it as we have done in previous sections. We report the 95% CIs based on the standard error of experimental and model results as well as the NW-CIs for the difference in experimental and model proportions. We also analyze the proportion of worse-than-random responses, as Maidenbaum et al. (2025) reports all participants were capable of performing the task better than random for all responses. We ran our model 100 times with randomized landmarks and locations, gathering the CED for each trial's responses. We also ran a parameter-matched version of the original snSLAM model to establish another comparison point.

7.1.1 Results

Figure 7.2 provides a visual comparison of MCED distributions for experimental and model results. The experimental and model results are strongly in agreement, having 95% CIs of 0.08 ± 0.01 and 0.08 ± 0.02 respectively. Both human participants and the model tend to respond at about the 92^{nd} percentile with fairly low error. Maidenbaum et al. (2025) additionally reports that no participant responses had a CED above 0.5; however, we make note that our model had a rate of CEDs above 0.5 of 6.25%. The increased rate of poor responses shows a statistically significant difference based on its NW-CI with a moderate effect size, $\Delta \in (0.04, 0.09)$ and $|h| = 0.51$, suggesting a worsened ability of the model to perform the task compared to human participants. The NW-CI and the Cohen's h for the difference in MCEDs, $\Delta \in (-0.02, 0.04)$ and $|h| = 0.01$, reveal a very narrow CI with no significant difference and a near-zero effect size. Since there is an increased rate of CEDs above 0.5 in the model compared to human participants despite a near-zero difference in average CEDs, we note that our model must also have a higher concentration of below-average CEDs than human participants. Both the CI and the Cohen's h reinforce that, despite some exceptionally poor responses, there is limited difference between the average positional accuracy of human participants and our model. Since we only have two data points, the mean and median Cohen's h are identical and not particularly informative; however for completeness, the mean and median absolute Cohen's h are 0.26. See Table A.3 in Appendix A for a complete report of confidence intervals and effect sizes. Overall, our results demonstrate that our model can be used as a dynamic memory system within a spiking neural SLAM framework, achieving human-like performance on a one-shot spatial memory task with an additional distractor condition.

Lastly, we compare to the results of the parameter-matched snSLAM model for the same experimental paradigm. Figure 7.3 shows the MCED for our model (STM+ITM SLAM) and the parameter-matched snSLAM compared to human experimental results. Both our model and the snSLAM model have good positional accuracy with the snSLAM model being slightly more accurate on average compared to both our model and human experimental data. Similarly, both models have a non-zero, albeit low, proportion of responses that are worse than random guessing—6.25% for our model and 4.75% for the snSLAM with absolute Cohen's h values of 0.51 and 0.44 respectively when compared to the human data. With respect to positional accuracy, our model is slightly more reflective of human-like positional accuracy than the snSLAM; our model's MCED showed an absolute Cohen's h of 0.01 in comparison to the absolute Cohen's h of 0.08 from the snSLAM. While both models have small effect sizes and no statistically significant difference in MCED when compared to the human data, our model is, qualitatively, slightly better than the original snSLAM, as the MCED is more consistent with human MCED. Overall, our model succeeds

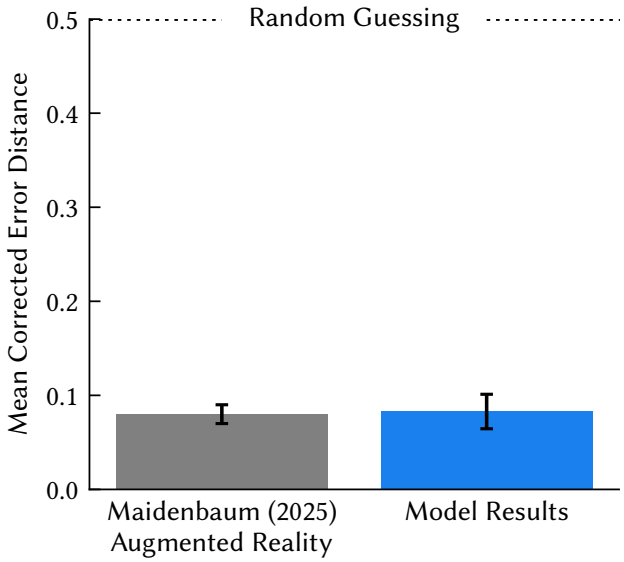


Figure 7.2: *Model and experimental spatial recall results.* Mean Corrected Error Distance (MCED) distributions for model and experimental results. Bars show the mean MCED for all participant and model results. Error bars show 95% confidence intervals using standard error (Model: $n = 400$, Maidenbaum: $n = 1800$).

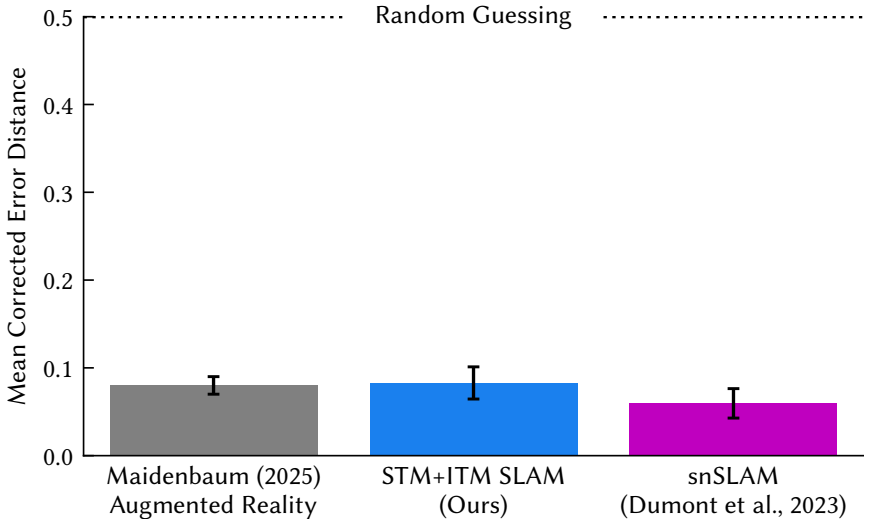


Figure 7.3: *A comparison of STM+ITM SLAM and snSLAM models to experimental spatial recall results.* Mean Corrected Error Distance (MCED) distributions for experimental, STM+ITM SLAM, and snSLAM results. Error bars show 95% confidence intervals using standard error (Maidenbaum: $n = 1800$, STM+ITM: $n = 400$, snSLAM: $n = 400$).

in augmenting the original snSLAM's map-learning capabilities by introducing the working memory and temporal integrator, resulting in a system more behaviourally consistent with human participants.

Chapter 8

Results Summary and Conclusion

8.1 Results Summary

Across serial recall, free recall, and spatial recall tasks, we observe that our model is qualitatively consistent with human participants. We see clear primacy effects, recency effects, and high positional accuracy in serial recall; recency effects and forward-biased conditional recall probability in free recall; and high positional accuracy in spatial recall. In addition to these characteristics, we find that the model recalls a proportion of list items consistent with that of human participants in both serial and free recall tasks and provides human-like responses for spatial recall tasks with respect to positional accuracy. We note, however, that there are a few areas in which the model performs worse than human participants, such as recalling the terminal items of lists less often for both serial and free recall and recalling a non-zero number of worse-than-random positions for spatial recall. Despite these deficits, model and human performance are qualitatively and quantitatively very similar. The similarity between our results and those of human participants suggests that our model is capable of exhibiting human-like memory. Analysis of effect size magnitudes strongly reinforce this interpretation.

Across our model's tasks, we examined 51 proportions as points of comparison: 12 for serial recall, 37 for free recall, and 2 for spatial recall. Of these 51 points, only 14 show statistically significant differences using 95% CIs. Despite their statistical significance, only 5 of the 14 differences, or 9.8% of all points of comparison, demonstrate non-small effect sizes: $0.41 < |h_M| < 0.65$. The remaining 91.2% of data points demonstrate small effect sizes, $0.002 < |h_S| < 0.31$, and no differences demonstrate large effect sizes. To gain better insight into the model's overall performance, we examine the distribution of effect size magnitudes, $|h|$, through their mean and median measures, $|\bar{h}|$ and $|\hat{h}|$ respectively.

Since we have no *a priori* knowledge as to what model would best fit the distribution of $|h|$, we construct 95% CIs for the mean and median by bootstrap sampling the set of all recorded $|h|$ values, H^+. We sample $|H^+|$ values from H^+ with replacement N times, where $|H^+|$ is the number of values in $H+$, and calculate the mean, $|\bar{h}|$, and median, $|\hat{h}|$, of each sampled set of values. We calculate the 2.5^{th} and 97.5^{th} percentile values from the sets of means and medians to construct the two-tailed 95% CI for each. For N=10,000, we find $|\bar{h}| \in (0.125, 0.203)$ and $|\hat{h}| \in (0.082, 0.189)$. Figure 8.1 provides a visualization for the distribution of $|h|$ values as well as the bootstrapped CIs for their mean and median. We see an evident concentration of small effect sizes as well as a some outlier values within the moderate effect size range. We also note that the 95% CI ranges for the mean and median magnitude of effect sizes fall comfortably within the range of small effect sizes.

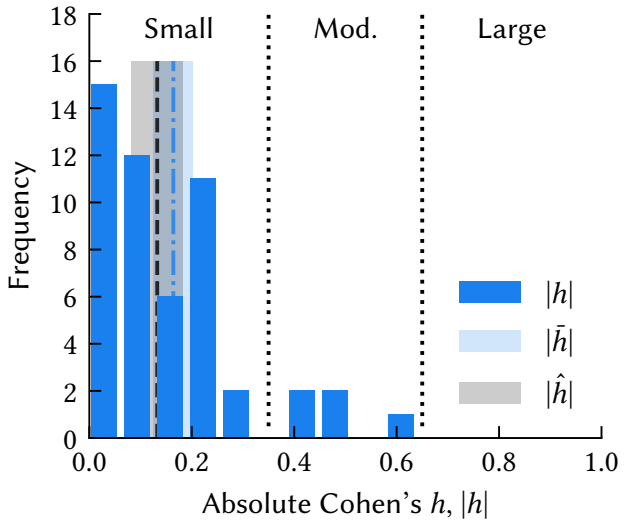


Figure 8.1: *Distribution of absolute effect sizes for results across tasks.* Bars show the frequency of absolute effect sizes for each bin. Shaded areas represent the 95% CI range of estimated values for the mean and median absolute effect sizes using bootstrapping, $|\bar{h}|$ and $|\hat{h}|$, respectively. Dashed and dot-dashed lines show the centre of the mean and median absolute effect size CIs, respectively.

8.2 Conclusion

Throughout this thesis, we have presented a novel model of short-term memory, a novel model of intermediate-term memory, and demonstrated their combined ability to realize

human-like behaviour in diverse experimental paradigms. Our model allows for a realization of short-term memory capacity and recency effects using a novel combination of projection-based dynamics and neural saturation, which we can characterize as an additional projection operation introducing non-linear dynamics reflective of the spiking neural dynamics. Additionally, our use of an auto-associative tensor product network demonstrates a novel application of local learning rules learning implicit associations within a spiking neural network. Unlike most of the models previously explored (e.g., the OSE, SSP-Attend, NAA, TCM, CMR, MHN, TEM, EntNet, and CUE models), we do not treat memory as discretely defined operations. We instead focus on its expression in a dynamical system. Further unlike the TCM and CMR, our model does not depend on stochastic processes for updating an internal state from which primacy and CRP effects arise. Instead, our model maintains an internal state, i.e., the temporal integrator, using dynamics similar to those implementing the short-term memory. Lastly, we complete all of our learning online, unlike most artificial neural network models (e.g., TEM, MHN, and EntNet models). Our emphasis on Hebbian learning processes helps to reinforce biological plausibility in a way that artificial neural networks depending on backpropagation cannot match. Overall, our model addresses many of the weaknesses of existing models, removing the need for discrete memory processing, orchestrated stochastic updating, and offline learning.

We embedded our memory models into larger models to handle input and signal routing for stimulus presentation and recall, and we applied the complete model to semantic memory and spatial memory tasks. We achieved performance comparable to the original human participants of each task and demonstrated domain generalization of our model—a necessity for understanding how large-scale spiking neural networks can realize human-like behaviour. Compared to prior spiking neural models—the snOSE, CUE, and snSLAM models—our model performed similarly in matching human data across all three memory tasks. Our model performed substantially better than the snOSE in serial recall while removing the orchestrated sequence of updates required to realize the TCM in the CUE; however, our model performed slightly worse than the CUE in both serial and free recall tasks. Our model also went a step further than both the snOSE and the CUE models by performing map-learning in a spatial memory task, demonstrating the generalizability of its architecture.

While our model makes strides in generalizable approaches to biologically plausible memory modelling, significant work remains. The inconsistencies of our model when compared to experimental data, such as the weakened ability to recall terminal list items, the reduced conditional recall probability for adjacent items, and the recall of worse-than-random positions for 6.25% of queries in spatial recall highlight significant room for improvement. While we performed local tuning of model parameters between tasks, we could further

improve model results by optimizing model parameters (e.g., η, g, r_m, etc.) and neural parameters (e.g., firing rates, biases, etc.). Since there were many interacting model parameters and long simulation times to acquire 100 results for each task, we performed only local searches, relying on original parameter estimates to establish a baseline. Exploring a larger parameter space and revisiting initial estimates—particularly the decay parameter (η) which was calculated based on existing literature—would most likely improve our model, becoming increasingly human-like in its serial and free recall performance. With respect to neural parameters, we rely on random sampling for neurons' preferred stimuli and firing rate biases. Consequently, we cannot guarantee that a given simulation will have a distribution of parameters that well-matches the underlying distributions, potentially leading to degraded task performance. Our relatively low dimensionality for SSPs may also lead to violations of our assumption that bound products will be approximately orthogonal. While we attempt to address the flags raised by neural parameters and SSP dimensionalities by running a large set of trials for our model, increasing dimensionality as well as the number of neurons in each population would help to prevent inopportune simulation initializations. We could also improve overall results be revisiting parameters within subnetworks used throughout our models, as pre-packaged HRR binding networks are optimized for unit-length vectors while the STM and ITM_p networks typically output a vector with a magnitude of $r \geq 1$.

With respect to future work, we note that this model leaves out long-term memory and its potential role in working memory tasks. Towards this, our ITM_L network creates a semi-stable representation of the task embedded within its auto-associative weights. We could stabilize these representations through recall-based consolidation mediated by the ITM_p network. The ITM_p network would progressively solidify associations from the ITM_L network through slower learning processes with access to both more neurons and modal specificity. We would also be interested in embedding this model into a more general network such that for any task of interest, the model would be identical in its connectivity.

Lastly, we express interest in adapting the ITM_L network to take on a form more consistent with a Modern Hopfield Network, as this would exponentially increase its capacity without strictly changing its ability to learn implicit associations. Unfortunately, methods on Modern Hopfield Networks learning high-dimensional, continuous-valued patterns fully in spiking neurons do not exist yet. Despite the limitations of the ITM_L as a tensor product network, it is sufficient for our use case, and this interest is primarily based on potential avenues for future work.

As a concluding remark, we note that our model provides a novel, general, and scalable way to implement both short-term and intermediate-term memory using dynamics realized in a biologically-plausible framework.

References

Aristotle. (1930). On Memory and Reminiscence (J. I. Beare, Trans.). In W. D. Ross (Ed.), *The Works of Aristotle* (Vol. 3). Clarendon Press. (Original work published 350 B.C.E.)

Bochner, S. (1959). *Lectures on Fourier Integrals*. Princeton University Press.

Brady, T. F., Konkle, T., & Alvarez, G. A. (2011). A review of visual memory capacity: Beyond individual items and toward structured representations. *Journal of Vision*, *11*(5), 4–4. https://doi.org/10.1167/11.5.4

Camina, E., & Güell, F. (2017). The Neuroanatomical, Neurophysiological and Psychological Basis of Memory: Current Models and Their Origins. *Frontiers in Pharmacology*, *8*, 438. https://doi.org/10.3389/fphar.2017.00438

Choo, F.-X. (2010, August). *The Ordinal Serial Encoding Model: Serial Memory in Spiking Neurons* [Master's thesis, University of Waterloo]. https://uwspace.uwaterloo.ca/items/d8c13ae2-daac-4cd1-885f-041fa94aa478

Chouinard, J. (2026). Neurally-plausible radial basis kernels using distributed Fourier embeddings. https://doi.org/10.48550/arXiv.2605.08458

Chouinard, J., & Eliasmith, C. (2026). A neural model of short-term and intermediate-term memory using neuro-symbolic representations for semantic and spatial memory tasks. *Proceedings of the Annual Meeting of the Cognitive Science Society*, *48*, 83–90.

Cicero. (1942). *De Oratore* (E. Sutton & H. Rackham, Trans.). Harvard University Press. (Original work published 55 B.C.E.)

Clopper, C. J., & Pearson, E. S. (1934). The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial. *Biometrika*, *26*(4), 404–413. https://doi.org/10.1093/biomet/26.4.404

Cohen, J. (2013). *Statistical Power Analysis for the Behavioral Sciences* (2nd ed.). Routledge. https://doi.org/10.4324/9780203771587

Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *The Behavioral and Brain Sciences*, *24*(1), 87–114, discussion 114–185. https://doi.org/10.1017/s0140525x01003922

Deese, J., & Kaufman, R. A. (1957). Serial effects in recall of unorganized and sequentially organized verbal material. *Journal of Experimental Psychology*, *54*(3), 180–187. https://doi.org/10.1037/h0040536

Dumont, N. S.-Y., & Eliasmith, C. (2020). Accurate representation for spatial cognition using grid cells. *Proceedings of the Annual Meeting of the Cognitive Science Society*, *42*, 2367–2373.

Dumont, N. S.-Y., Furlong, P. M., Orchard, J., & Eliasmith, C. (2023). Exploiting semantic information in a spiking neural SLAM system. *Frontiers in Neuroscience*, *17*, 1190515. https://doi.org/10.3389/fnins.2023.1190515

Ebbinghaus, H. (1913). *Memory; A Contribution to Experimental Psychology* (H. Ruger, Trans.). Teachers College, Columbia University. (Original work published 1885)

Eliasmith, C., & Anderson, C. (2003). *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. MIT Press.

Eliasmith, C. (2013). *How to Build a Brain: A Neural Architecture for Biological Cognition*. Oxford University Press.

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., & Rasmussen, D. (2012). A Large-Scale Model of the Functioning Brain. *Science*, *338*(6111), 1202–1205. https://doi.org/10.1126/science.1225266

Farrell, S., & Lewandowsky, S. (2004). Modelling transposition latencies: Constraints for theories of serial order memory. *Journal of Memory and Language*, *51*(1), 115–135. https://doi.org/10.1016/j.jml.2004.03.007

Frady, E. P., Kleyko, D., Kymn, C. J., Olshausen, B. A., & Sommer, F. T. (2021). *Computing on Functions Using Randomized Vector Representations*. arXiv: 2109.03429.

Golomb, J. D., Peelle, J. E., Addis, K. M., Kahana, M. J., & Wingfield, A. (2008). Effects of adult aging on utilization of temporal and semantic associations during free and serial recall. *Memory & Cognition*, *36*(5), 947–956. https://doi.org/10.3758/MC.36.5.947

Gosmann, J., & Eliasmith, C. (2021). CUE: A unified spiking neuron model of short-term and long-term memory. *Psychological Review*, *128*(1), 104–124. https://doi.org/10.1037/rev0000250

Gosmann, J., Voelker, A. R., & Eliasmith, C. (2017). A Spiking Independent Accumulator Model for Winner-Take-All Computation. *Proceedings of the Annual Meeting of the Cognitive Science Society*, *39*, 2125–2130. https://escholarship.org/uc/item/01z51564

Goto, A. (2022). Synaptic plasticity during systems memory consolidation. *Neuroscience Research*, *183*, 1–6. https://doi.org/10.1016/j.neures.2022.05.008

Hebb, D. (1961). Distinctive features of learning in the higher animal. *Proceedings of the Council for International Organizations of Medical Sciences*, 37–46.

Henaff, M., Weston, J., Szlam, A., Bordes, A., & LeCun, Y. (2016). Tracking the World State with Recurrent Entity Networks. https://doi.org/10.48550/arXiv.1612.03969

Henson, R. N. A., Norris, D. G., Page, M. P. A., & Baddeley, A. D. (1996). Unchained Memory: Error Patterns Rule out Chaining Models of Immediate Serial Recall. *The Quarterly Journal of Experimental Psychology A*, *49*(1), 80–115. https://doi.org/10.1080/027249896392810

Hilton, C., Kapaj, A., & Fabrikant, S. I. (2023). Landmark Sequence Learning from Real-World Route Navigation and the Impact of Navigation Aid Visualisation Style. *Journal of Cognition*, *6*(1), 41. https://doi.org/10.5334/joc.307

Howard, M. W., & Kahana, M. J. (1999). Contextual variability and serial position effects in free recall. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *25*(4), 923–941. https://doi.org/10.1037/0278-7393.25.4.923

Howard, M. W., & Kahana, M. J. (2002). A Distributed Representation of Temporal Context. *Journal of Mathematical Psychology*, *46*(3), 269–299. https://doi.org/10.1006/jmps.2001.1388

Jahnke, J. C. (1968). Delayed recall and the serial-position effect of short-term memory. *Journal of Experimental Psychology*, *76*(4, Pt.1), 618–622. https://doi.org/10.1037/h0025692

Jenkins, J. J., & Russell, W. A. (1952). Associative clustering during recall. *The Journal of Abnormal and Social Psychology*, *47*(4), 818–821. https://doi.org/10.1037/h0063149

Kahana, M. J. (2017). 2.11 - Memory Search. In *Learning and Memory: A Comprehensive Reference* (2nd ed., pp. 181–200). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-809324-5.21038-9

Kamiński, J. (2017). Intermediate-Term Memory as a Bridge between Working and Long-Term Memory. *The Journal of Neuroscience*, *37*(20), 5045–5047. https://doi.org/10.1523/JNEUROSCI.0604-17.2017

Kamiński, J., Sullivan, S., Chung, J. M., Ross, I. B., Mamelak, A. N., & Rutishauser, U. (2017). Persistently active neurons in human medial frontal and medial temporal lobe support working memory. *Nature Neuroscience*, *20*(4), 590–601. https://doi.org/10.1038/nn.4509

Komer, B., Stewart, T. C., Voelker, A. R., & Eliasmith, C. (2019). A neural representation of continuous space using fractional binding. *Proceedings of the Annual Meeting of*

the Cognitive Science Society, *41*, 2038–2043. https://escholarship.org/uc/item/3zz346g1

Laczó, M., Wiener, J. M., Kalinova, J., Matuskova, V., Vyhnalek, M., Hort, J., & Laczó, J. (2021). Spatial Navigation and Visuospatial Strategies in Typical and Atypical Aging. *Brain Sciences*, *11*(11), 1421. https://doi.org/10.3390/brainsci11111421

Lu, T., Voelker, A. R., Komer, B., & Eliasmith, C. (2019). Representing Spatial Relations with Fractional Binding. *Proceedings of the Annual Meeting of the Cognitive Science Society*, *41*. https://escholarship.org/uc/item/3m52p2nx

Maidenbaum, S., Kremen, V., Sladky, V., Miller, K., Gompel, J. V., Worrell, G. A., & Jacobs, J. (2025). Improved spatial memory for physical versus virtual navigation. *Journal of Neural Engineering*, *22*(4), 046014. https://doi.org/10.1088/1741-2552/ade6aa

Murdock, B. B. (1962). The serial position effect of free recall. *Journal of Experimental Psychology*, *64*(5), 482–488. https://doi.org/10.1037/h0045106

Newcombe, R. G. (1998). Interval estimation for the difference between independent proportions: Comparison of eleven methods. *Statistics in Medicine*, *17*(8), 873–890. https://doi.org/10.1002/(SICI)1097-0258(19980430)17:8⟨873::AID-SIM779⟩3.0.CO;2-I

Penzkofer, A., Furlong, M., Eliasmith, C., & Bulling, A. (2025). A Cognitively Plausible Model of Visual Working Memory. *Proceedings of the Annual Meeting of the Cognitive Science Society*, *47*, 2330–2336. https://escholarship.org/uc/item/3928d5s4

Plate, T. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, *6*(3), 623–641. https://doi.org/10.1109/72.377968

Polyn, S. M., Norman, K. A., & Kahana, M. J. (2009). A context maintenance and retrieval model of organizational processes in free recall. *Psychological Review*, *116*(1), 129–156. https://doi.org/10.1037/a0014420

Poo, M.-m., Pignatelli, M., Ryan, T. J., Tonegawa, S., Bonhoeffer, T., Martin, K. C., Rudenko, A., Tsai, L.-H., Tsien, R. W., Fishell, G., Mullins, C., Gonçalves, J. T., Shtrahman, M., Johnston, S. T., Gage, F. H., Dan, Y., Long, J., Buzsáki, G., & Stevens, C. (2016). What is memory? The present state of the engram. *BMC Biology*, *14*(1), 40. https://doi.org/10.1186/s12915-016-0261-6

Rahimi, A., & Recht, B. (2007). Random Features for Large-Scale Kernel Machines. In J. Platt, D. Koller, Y. Singer, & S. Roweis (Eds.), *Advances in Neural Information Processing Systems* (Vol. 20). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., Gruber, L., Holzleitner, M., Pavlović, M., Sandve, G. K., Greiff, V., Kreil, D., Kopp, M., Klambauer,

G., Brandstetter, J., & Hochreiter, S. (2020). Hopfield Networks is All You Need. https://doi.org/10.48550/arXiv.2008.02217

Reimann, S. (2025). Memory States From Almost Nothing: Representing and Computing in a Nonassociative Algebra. *Neural Computation*, *37*(6), 1154–1170. https://doi.org/10.1162/neco_a_01755

Salvucci, D. D. (2001). An integrated model of eye movements and visual encoding. *Cognitive Systems Research*, *1*(4), 201–220. https://doi.org/10.1016/S1389-0417(00)00015-2

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, *46*(1-2), 159–216. https://doi.org/10.1016/0004-3702(90)90007-M

Snow, M., & Orchard, J. (2022). Biological Softmax: Demonstrated in Modern Hopfield Networks. *Proceedings of the Annual Meeting of the Cognitive Science Society*, *44*, 505–511.

Spens, E., & Burgess, N. (2024). A generative model of memory construction and consolidation. *Nature Human Behaviour*, *8*(3), 526–543. https://doi.org/10.1038/s41562-023-01799-z

The MathWorks Inc. (2026). *System identification toolbox version: 26.1 (r2026a)*. Natick, Massachusetts, United States. https://www.mathworks.com

Voelker, A. R. (2020). *A short letter on the dot product between rotated Fourier transforms*. arXiv: 2007.13462.

Voelker, A. R., Gosmann, J., & Stewart, T. C. (2017). Efficiently sampling vectors and coordinates from the n-sphere and n-ball. https://doi.org/10.13140/RG.2.2.15829.01767/1

Wamsley, E. J. (2019). Memory Consolidation during Waking Rest. *Trends in Cognitive Sciences*, *23*(3), 171–173. https://doi.org/10.1016/j.tics.2018.12.007

Whittington, J. C. R., Muller, T. H., Mark, S., Chen, G., Barry, C., Burgess, N., & Behrens, T. E. J. (2020). The Tolman-Eichenbaum Machine: Unifying Space and Relational Memory through Generalization in the Hippocampal Formation. *Cell*, *183*(5), 1249–1263.e23. https://doi.org/https://doi.org/10.1016/j.cell.2020.10.024

Wilson, M. A., & McNaughton, B. L. (1994). Reactivation of hippocampal ensemble memories during sleep. *Science*, *265*(5172), 676–679. https://doi.org/10.1126/science.8036517

APPENDIX

Appendix A

Results Tables

The following tables provide a breakdown of the CP-CIs for model and experimental results—β_M and β_E respectively—the NW-CIs for their differences—Δ—and their corresponding effect sizes—Cohen's h. All confidence intervals reflect 95% confidence measures. Table A.1 summarizes the results of the serial recall task; Table A.2 summarizes the results of the free recall task; and Table A.3 summarizes the results of the spatial recall task. Note that since Corrected Error Distance is not formed from a set of binomial variables, we use the central limit theorem and approximate its CIs using a 95% Gaussian CI.

Table A.1: *Summary of Data Analyses for Serial Recall Results.* Statistically significant differences and non-small effect sizes bolded.

Metric	Index	β_M	β_E	Δ	h
Recall Proportion	N	(0.325, 0.386)	(0.352, 0.415)	(-0.071, 0.014)	0.115
Serial Position Curve	S1	(0.467, 0.669)	(0.573, 0.769)	(-0.237, 0.028)	-0.222
	S2	(0.212, 0.400)	(0.346, 0.553)	**(-0.276, -0.013)**	-0.307
	S3	(0.152, 0.325)	(0.141, 0.315)	(-0.106, 0.127)	0.027
	S4	(0.169, 0.347)	(0.150, 0.326)	(-0.099, 0.139)	0.049
	S5	(0.118, 0.281)	(0.082, 0.233)	(-0.062, 0.149)	0.118
	S6	(0.118, 0.281)	(0.194, 0.382)	(-0.208, 0.028)	-0.216
	S7	(0.160, 0.336)	(0.222, 0.415)	(-0.196, 0.052)	-0.162
	S8	(0.169, 0.347)	(0.240, 0.437)	(-0.207, 0.044)	-0.184
	S9	(0.266, 0.462)	(0.457, 0.664)	**(-0.331, -0.063)**	**-0.409**
	S10	(0.379, 0.582)	(0.489, 0.693)	(-0.247, 0.025)	-0.229
Positional Accuracy	T0	(0.822, 0.896)	(0.778, 0.858)	(-0.095, 0.012)	0.115

Table A.2: *Summary of Data Analyses for Free Recall Results.* Statistically significant differences and non-small effect sizes bolded.

Metric	Index	β_M	β_E	Δ	h
Recall Proportion	N	(0.347, 0.402)	(0.365, 0.386)	(-0.030, 0.028)	-0.001
Serial Position Curve	S1	(0.204, 0.389)	(0.233, 0.300)	(-0.062, 0.125)	0.055
	S2	(0.143, 0.314)	(0.168, 0.228)	(-0.053, 0.118)	0.057
	S3	(0.143, 0.314)	(0.159, 0.218)	(-0.042, 0.128)	0.082
	S4	(0.135, 0.303)	(0.155, 0.213)	(-0.047, 0.121)	0.069
	S5	(0.143, 0.314)	(0.148, 0.206)	(-0.032, 0.139)	0.112
	S6	(0.204, 0.389)	(0.195, 0.258)	(-0.022, 0.164)	0.148
	S7	(0.177, 0.357)	(0.156, 0.215)	(-0.006, 0.164)	0.183
	S8	(0.266, 0.462)	(0.223, 0.289)	**(0.011, 0.207)**	0.227
	S9	(0.331, 0.533)	(0.329, 0.402)	(-0.035, 0.164)	0.133
	S10	(0.428, 0.631)	(0.586, 0.659)	(-0.197, 0.008)	-0.189
	S11	(0.569, 0.761)	(0.845, 0.895)	**(-0.301, -0.113)**	**-0.491**
	S12	(0.697, 0.865)	(0.963, 0.987)	**(-0.277, -0.117)**	**-0.649**
First Recall Curve	F1	(0.049, 0.176)	(0.034, 0.067)	**(0.003, 0.127)**	0.199
	F2	(0.003, 0.054)	(0.002, 0.015)	(-0.008, 0.049)	0.049
	F3	(0.006, 0.085)	(0.001, 0.012)	**(0.004, 0.080)**	0.217
	F4	(0.000, 0.054)	(0.005, 0.022)	(-0.015, 0.043	-0.014
	F5	(0.006, 0.085)	(0.002, 0.015)	**(0.003, 0.079)**	0.197
	F6	(0.006, 0.085)	(0.011, 0.033)	(-0.014, 0.065)	0.065
	F7	(0.006, 0.085)	(0.008, 0.028)	(-0.009, 0.069)	0.097
	F8	(0.006, 0.085)	(0.011, 0.033)	(-0.014, 0.065)	0.065
	F9	(0.011, 0.099)	(0.029, 0.061)	(-0.033, 0.057)	-0.014
	F10	(0.042, 0.164)	(0.090, 0.138)	(-0.072, 0.053)	-0.075
	F11	(0.143, 0.314)	(0.159, 0.218)	(-0.043, 0.128)	-0.082
	F12	(0.285, 0.483)	(0.440, 0.516)	(-0.194, 0.007)	-0.198
Conditional Response Curve	C-6	(0.030, 0.102)	(0.026, 0.044)	(-0.010, 0.093)	0.126
	C-5	(0.017, 0.058)	(0.028, 0.045)	(-0.028, 0.057)	-0.003
	C-4	(0.019, 0.078)	(0.034, 0.054)	(-0.030, 0.065)	0.010
	C-3	(0.070, 0.148)	(0.059, 0.084)	(-0.016, 0.113)	0.126
	C-2	(0.068, 0.137)	(0.096, 0.013)	(-0.061, 0.068)	-0.031
	C-1	(0.098, 0.195)	(0.223, 0.269)	**(-0.167,-0.014)**	-0.290
	C1	(0.183, 0.309)	(0.357, 0.418)	(-0.225, -0.043)	-0.308
	C2	(0.038, 0.120)	(0.082, 0.121)	(-0.070, 0.047)	-0.089
	C3	(0.044, 0.126)	(0.039, 0.068)	(-0.014, 0.100)	0.114
	C4	(0.058, 0.160)	(0.030, 0.057)	**(0.014, 0.141**	0.249
	C5	(0.025, 0.115)	(0.015, 0.036)	**(0.003, 0.107)**	0.200
	C6	(0.045, 0.160)	(0.005, 0.017)	**(0.042, 0.162**	**0.430**

Table A.3: *Summary of Data Analyses for Spatial Recall Results.* Statistically significant differences and non-small effect sizes bolded.

Metric	Index	β_M	β_E	Δ	h
MCED	μ	(0.064, 0.101)	(0.070, 0.090)	(-0.024, 0.036)	0.010
"Bad" Responses	b	(0.041, 0.253)	(0.000, 0.001)	**(0.043, 0.091)**	**0.505**