# Learning adaptive gating strategies underlies effective working memory across biological and artificial networks

By

Aneri Soni

B.S., Cornell University, 2018

M.S., Brown University, 2021

Thesis

Submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in the Department of Neuroscience at Brown University

Providence, Rhode Island

February 2025

This dissertation by Aneri Soni is accepted in its present form by the Department of Neuroscience as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date _____ _____

Dr. Michael J. Frank, Advisor

Recommended to the Graduate Council

Date _____ _____

Dr. Stephanie Jones, Committee Chair

Date _____ _____

Dr. David Sheinberg, Reader

Date _____ _____

Dr. Thomas Serre, Reader

Date _____ _____

Dr. Susanne Jaeggi, Outside Reader

Approved by the Graduate Council

Date _____ _____

Thomas A. Lewis, Dean of the Graduate School

# Curriculum Vitae

**Aneri Soni**

https://www.linkedin.com/in/anerivsoni/

## Education

**PhD**, Brown University — Feb 2025
Neuroscience — Providence, Rhode Island — GPA: 4.0

**MS**, Brown University — May 2021
Neuroscience — Providence, Rhode Island — GPA: 4.0

**BA**, Cornell University — May 2018
Psychology Major, Math Minor — Ithaca, New York — GPA: 3.9
Cum Laude and With Honors

## Selected Publications

- In Prep for ICLR 2025: Frontostriatal Gating in Transformers when trained on Working Memory Tasks; **Aneri Soni**$\hat{*}$, Aaron Traylor$\hat{*}$, Jack Merullo, Michael J. Frank, Ellie Pavlick
- Published at eLife: Adaptive Chunking improves effective working memory capacity in a prefrontal cortex and basal ganglia circuit; **Aneri Soni** and Michael Frank, 2024
- Submitted at arXiv 2021: KuraNet: systems of coupled oscillators that learn to synchronize; Matthew Ricci, Minju Jung, Yuwei Zhang, Mathieu Chalvidal, **Aneri Soni**, Thomas Serre $\hat{*}$ indicates equal contribution

## Research Experience

Research Fellow - Neuroscience — Brown University
Frank Lab, Providence, RI — September 2019 - Present

- **Biomimetic Gating in Transformers for Language:**
- Goal: 1) Improve learning challenges faced by Transformers 2) Mechanistic solution (based on gating in working memory)
- Trained and tested models (including pretraining and finetuning) using GPUs; continuous integration for codebase.
- Achieved a 15.7x reduction in error rate for generalization to new symbols.
- Reduced training time by 35% with the mechanistic solution (path-patching to test mechanistic interpretability).

- **Reinforcement Learning NNs using lossy compression for visual tasks:**
- Goal: Design architecture for lossy compression into visual working memory ML models based on human data.
- Designed and trained a reinforcement learning ML model inspired by dopamine pathways.
- Model results made novel predictions about cognitive deficits in patient populations (ex. Parkinson's, schizophrenia).
- Achieved a 60% increase in efficient resource management and an 18% increase in accuracy with the new chunking model.
- Concluded that working memory performance is dependent on efficient storage rather than capacity.
- General Experience: Lead the intellectual trajectory of projects; Mentor younger scientists; collaborate across all levels.

Research Fellow - Computer Vision — Brown University
Serre Lab, Providence, RI — September 2018 - September 2020

- Developed an artificial NN of nonlinear coupled oscillators (reservoir computing/recurrent NN) using PyTorch and GPUs.
- Designed the NN for visual reasoning and image segmentation (computer vision tasks).
- Interpretable neural networks: Tested ANNs for similarity to biological brain features using PyTorch/Keras and AWS; ImageNet.

Undergraduate Research Fellow — Cornell University
The Experimental Epistemology Laboratory — Ithaca, NY — August 2016 - May 2018

- Honors thesis: Analyzed resting state EEG to predict performance and resilience in visual tasks.
- Utilized Fractal Dimension analysis and Fourier Analysis (MATLAB, R).

Undergraduate Research Fellow — Cornell University
Computational Physiology Laboratory — Ithaca, NY — January 2015 - May 2016

- Analyzed brain slices to study neuronal activation during fear conditioning; modality – olfaction

**Selected Conferences and Talks**

- Invited Talk at Control Processes 2022: Chunking in a PFC-BG neural network as a strategy to overcome working memory capacity constraints; **Aneri Soni** and Michael Frank
- Chalk Talk at ICoN Grant Meeting, 2021: PFC and Working Memory

- Guest Lecturer, Computational Neuroscience Class (Brown University), 2020, **Aneri Soni** and Abdullah Rashed
- Poster at Cosyne Conference 2020: Kura-Net: Exploring systems of coupled oscillators with deep learning; Matthew Ricci, **Aneri Soni**, Thomas Serre, Yuwei Zhang, Minju Jung
- Invited Talk at MIT Brain and Cognitive Sciences Dicarlo Lab 2020: Interpretable Neural Networks: Brainscore; **Aneri Soni**, Pachaya Sailamul, Thomas Serre
- Poster Presentation at Cornell University Honors Thesis Symposium, 2018; *Can Resting State EEG Predict Performance in Visual Tasks?* **Aneri Soni** and Shimon Edelman
- Talk at Cornell University Honors Thesis Symposium 2018; *Can Resting State EEG Predict Performance in Visual Tasks?* **Aneri Soni** and Shimon Edelman

**Leadership and Teaching Experience** Research Mentor — Brown University
Frank Lab and Serre Lab — January 2019 - December 2024

- Mentored multiple undergraduate and graduate researchers

Conference Committee Member — Brown University
Brown Unconference — March 2020 - July 2020

- Organized first Brown Unconference
- Created outreach material, helped manage technology decisions, and event planning
- Moderated presentation sessions
- The conference had around 120 presenters, keynote speakers, and networking sessions

Teaching Assistant — Brown University
Neuroscience Department — January 2020 - May 2020

- TA for Graduate Statistics and Python Class (Neur2060)
- Led weekly in-person and virtual TA sessions

Student Speaker Committee Representative — Brown University
Neuroscience Department — June 2019 - June 2020

- Voted to be the graduate representative
- Solicit suggestions for speakers
- Attend meetings and organize speaker selection
- Promote underrepresented faculty as guests
- Gather feedback from prior graduate students
- Hosted speaker Yael Niv (Princeton)

Table Organizer — Brown University
Brain Week and Panels — September 2019 - September 2020

- Organize Tables for Brain Week
- Public outreach

Journal Group Organizer — Brown University
Machine Learning and Neuroscience — 2019-2021

- Select papers, organize meetings, and lead paper discussion

Senior Study Group Leader — Cornell University
Biology Department — January 2016 - May 2016

- Train other study group leaders
- Teach weekly classes of groups of 10 undergraduate students
- Attend weekly meetings and training sessions

Study Group Leader — Cornell University
Biology Department — August 2015 - December 2015

- Teach weekly classes of groups of 10 undergraduate students
- Attend weekly meetings and training sessions

E-board Member — Cornell University
Reach Tutoring — September 2015 - May 2018

- Organize and train tutors (Around 50 tutors)
- Maintain communication with schools in the community and continue community outreach
- Weekly meetings with other e-board members

**Awards and Grants**

- ICoN T32 Grant (2020-2022)
- NSF honorable mention (2020)
- Member of National Society of Collegiate Scholars (NSCS)
- Einhorn Discovery Grant 2017
- University College of Arts and Sciences Undergraduate Research Grant 2017
- Cornell University College of Arts and Science Summer Experience Grant 2016 and 2017
- Cornell Entrepreneurship Grant 2016 and 2017

**Volunteer**

School Visit — Brown University
Neuroscience Department — January 2019 - April 2019

- Visit local schools to educate about the brain

Researcher — Harvard University/University of Puerto Rico
VMove Public Health Initiative

- Literature review on barriers to exercise
- Manage social media presence
- Create website material
- Contact doctors and hospitals

Volunteer — Cayuga Medical Center
Outpatient Department — February 2017- May 2017

- Assist nurses and patients

Torch and Laurel Mentor
National Society of Collegiate Scholars — September 2015 - February 2016

- Advise and mentor high school students on college applications, essays, and majors

Tutor — Cornell University
Reach Tutoring — September 2015 - May 2018

- Tutor elementary children
- Attend training sessions

# Acknowledgments

I am deeply appreciative of the opportunity to be in the position to pursue an advanced degree in a field I love.

Thank you to my whole committee - Stephanie, David, Thomas, my outside reader Susanne, and of course my advisor Michael. Thanks to the whole committee for many fruitful discussions and continued support and enthusiasm for my work and personal growth.

To Michael, thank you for being an awesome advisor. Thank you for helping me grow as a scientist and shaping me into the researcher I am today. Doing research with you has been so much fun and I really enjoyed learning about science and the brain from you. I appreciate your patience and support as I navigated motherhood and graduate student life. Thank you to the rest of the lab members - it has been so much fun to work, hang out, chat, do journal clubs, and become friends with you guys.

Thank you to Thomas for kicking off my career in deep learning and to Ellie for being an amazing collaborator and helping me delve into the field of LLMs.

I have met so many wonderful scientists and researchers at Brown University and I want to thank the wonderful community that the Neuroscience department has worked to build. It really makes doing science special and fun. A special shout out to the older graduate students (too many to name) that connected with me and were role models and mentors during this process.

A special thanks to my lovely cohort - I've had so much fun learning and growing with you all. From classes to neuropracticum and to chatting on the third floor after seminars, it has been great.

To the neuroscience admin, thank you! Carol - for being so awesome and helpful from the first day of recruitment all the way to hitting submit on this thesis. Thank you Rosanna and all the other members that help make things run smoothly.

I want to acknowledge all my former teachers and educators. From preschool up to now, each has played an important role in shaping my curiosity, my interests, and my skills. A special thanks to my advisors at Cornell University David Smith, Shimon Edelman, and Thomas A. Cleland who all played an important role in starting my research career and in my decision to pursue graduate research.

I could not have done this without the support and love of my family. My parents who have sacrificed a lot to move to the United States and give us a more comfortable life and better education. To my sister who is always a supporter. To my partner in crime and husband, Sean. I am so glad to have you on my side for all our adventures. To my two wonderful kids, Hari and Arjun. You both make me so much stronger and happier. I appreciated your enthusiasm as you listened and watched me present parts of my talk over and over. Hari - for all your questions and Arjun - for asking me to practice the color wheel task again because you liked watching all the animations. You both were probably the biggest stressors as well as the biggest stress relievers through this process. I hope one day you guys can pursue your dreams and follow your passions.

# Contents

# List of Figures

# List of Tables

# Introduction

.

## 0.1 Working Memory

Working memory (WM) is a key cognitive function of the human brain that allows a person to navigate everyday life. Tasks like having a conversation, doing mental math, or even playing a soccer game require a functioning working memory.

A popular definition for working memory is "the few temporarily active thoughts [...] used in mental tasks such as language comprehension" (Cowan, 2010). WM is required to maintain objects/items that are no longer present in the environment, but the brain maintains for a short period of time. Usually the information kept in working memory is used in language compression, to inform decisions or choose the next action. The term working memory itself was originally coined by (Miller et al., 1960). Researchers have come a long way and we now better understand many aspects of working memory, some of which will be discussed in this introduction.

From a neuroscience perspective, working memory has long been regarded as "maintaining information over some delay" accompanied, by the popular figure by Funahashi 1989.

The conception that working memory is only about maintaining information is incomplete and research in cognitive science has shown that it is only one part of the process.

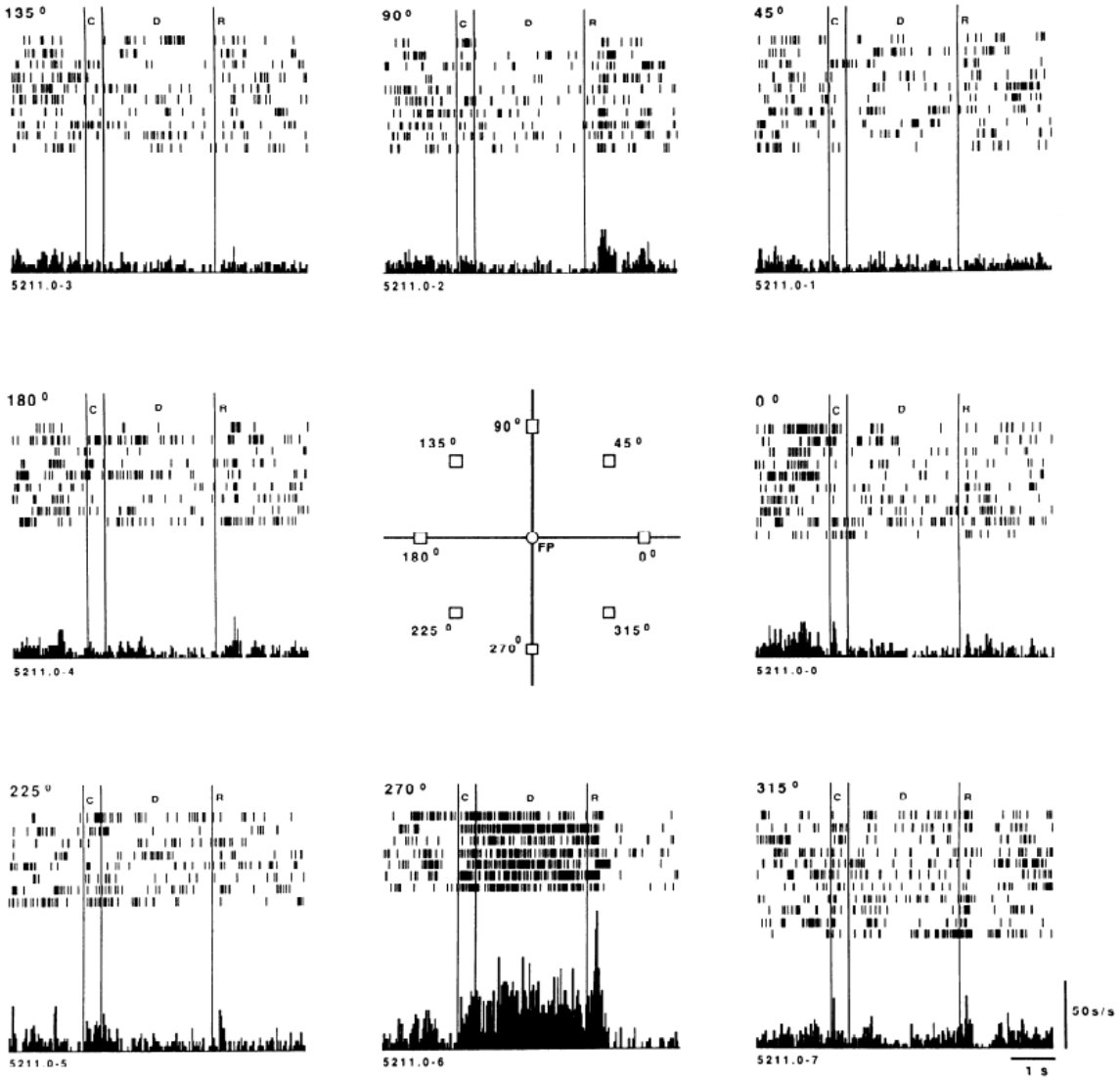



Figure 1: **Maintaining Information Over Time** (Adapted from Funahashi 1989) This shows different neurons maintaining infromation over some delay period where the original stimulus is not present. The neurons maintain information based on if their preferred degree of orientation matches that of the stimulus.

Here I will highlight the other relevant and key components of working memory and seek to shift away from just associating working memory with maintenance.

I break up working memory function into 3 components: **input gating**, **maintenance**, and **output gating**. Input gating requires the decision of what information from the environment is relevant. What should be remembered? Just as importantly, input gating is the process of placing the information in working memory and deciding how to store that information in a way that each item is separable. Usually, separate neural populations hold information about each item. The second component, maintenance, encompasses mechanisms that support persistence of information through the delay period. This also includes deciding which pieces of information should be continued to maintained, and which should be overwritten by other newer (and possibly more relevant) information. Lastly is output gating. Ultimately, the information that is stored is intended to guide decision making and therefore needs to be accessed. Output gating consists of deciding what information needs to be accessed and how (what form and where to ensure separation from other items) from working memory it can be read.

Whether it be biological (ex. human) or artificial, the agent needs to learn to solve these different components to have a successful working memory. It needs to learn what is relevant, where information should be stored, when it should be updated, and how to re-access the correct information. In situations and tasks involving working memory, only the resulting action (influenced by working memory) is rewarded/punished. Therefore intervening gating decisions do not get individual feedback. This results in a difficult credit assignment problem where the agent has to disentangle which gating decisions leading to action and reward were correct/incorrect. For example information may have been properly gated into working memory, but if it was overwritten or perhaps the wrong information was output gated, the improper action would be taken, leading to no reward despite proper input gating.

This learning process and how it connects with the rest of the existing working memory

literature is key to better understanding working memory. Currently, this connection is not fully understood or explored and here I aim to bridge that gap. In this thesis, I explore input and output gating operations and how they can be learned over time in the face of difficult credit assignment challenges. Better understanding the separate components for working memory and the learning problem will allow us to explore individual differences in working memory, and even working memory deficits in patients.

I compare the computational principles of working memory and performance on tasks across biological and multiple artificial networks. This serves to help 1) build computational models that serve as tools to better understand human working memory and 2) improve our understanding of artificial neural networks (specifically transformers) and to help train them better.

## 0.2 Working Memory Capacity Constraints

We know that working memory is limited (if you need convincing, try a long mental math problem). What are the factors contributing to working memory capacity? I present 3 ideas here and expand on them below.

1) A classical account for limited working memory is that the brain has a limited number of neurons and therefore limited storage for information. There is a lot of work (slots and resources) to discuss how information may be stored in working memory. Further, there is also work to discuss information compression and possible efficient encoding strategies.

2) A less studied idea is how role addressability and variable binding (a key property of working memory) is related to capacity.

3) Lastly, studies show that an individual's working memory capacity is linked with his/her ability to filter our irrelevant information (Cowan and Morey, 2006; McNab and Klingberg, 2008). This suggests that the filtering mechanism and input gating step is critical to the capacity.

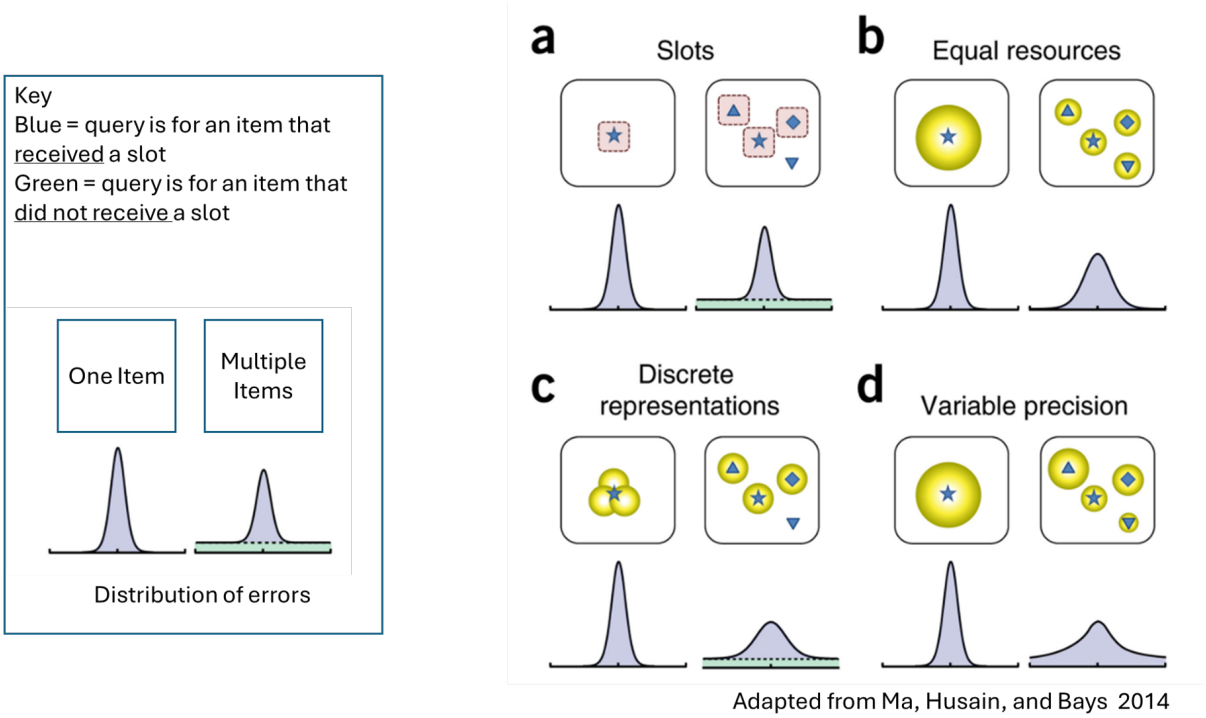



Figure 2: **Slots vs. Resources** (Adapted from Ma, Husain, and Bays 2014) The key gives the layout of the figures in panels a-d. a) Slots model, where there is a discrete cut off for the number of stimuli that can be held in memory b) equal resource model, where all items can receive space in memory, at the cost of lowered precision of recall for each item c) discrete representations, a hybrid model of the slots and resources where the resource can be pulled for increased precision but there is a discrete cut off and items are left out of memory d) variable precision, where all items can be given space in memory (again at the cost of precision) but also resources can be flexibly allocated between items based on importance: more important items can receive more resources and therefore have higher precision during recall.

### 0.2.1 Neural Populations and Form of Working Memory

The most well studied idea about capacity limit is the limited neural populations account and associated theories. These theories delve into the form of the limitation and what happens when capacity is exceeded.

There is a long-standing debate about the form of the capacity constraint of working memory. While some researchers focus on time-limit constraints, others focus on item-limit constraints. For the purpose here, I focus on the item-limit body of work, which itself is very large. Researchers exploring item-limits have historically formed into two camps:

slots (Cowan, 2010; Gilchrist et al., 2008; Zhang and Luck, 2008), and resources (Ma et al., 2014; Bays and Husain, 2008). The "slots" theory states there is a discrete limit to the number of items that can be in working memory. Once the number of items presented exceeds the capacity, some items are excluded from memory. Early work suggests that the discrete limit to human working memory is 7 +/- 2 (Miller 1956), while more current work has revised this capacity to be 3-5 items Cowan (2010); Gilchrist et al. (2008). The "resource" theory states that there is a continuous and flexibly divisible resource that can accommodate any number of items. However, since there is still a limit to the resource, remembering more items requires the resource to be divided into even smaller amounts and therefore comes at the cost of lower recall precision (more error) for each item. When looking at experimental data that aims to distinguish between these two theories, error distributions are widely used. Error distributions are generally centered around 0 (no error) and have tails that indicate answers that are completely incorrect or the opposite of the correct answer. When the environment exceeds the discrete limit capacity (i.e. more than 3-4 items), the slots error distribution is expected to be some gaussian-like distribution (for items in memory) combined with a uniform distribution that represents guessing for the items not in memory. The resource theory would predict that as the number of items in memory increases, the variance of the gaussian-like distribution would increase.

A sample task that is used is to study capacity constraint is known as the color wheel task: participants are given bars of various colors and orientations and after a short delay, the participant is asked to report the correct color by clicking on a continuous color wheel, given a bar of certain orientation. The continuous report is critical because researchers can then calculate the difference between target color and actual response from the participants and create the error distributions. This stimulus requires linking the orientation and color information together and therefore requires variable binding. Further, the orientation is used as a probe and therefore can be used to access other

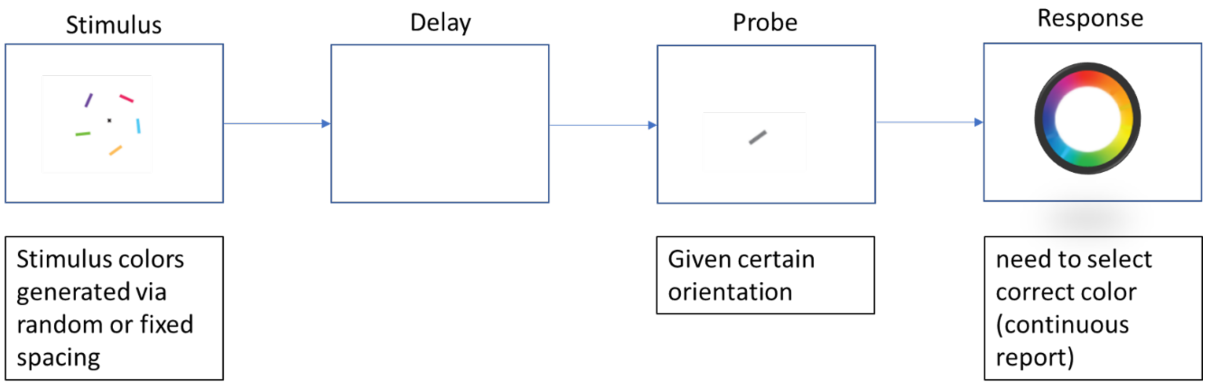



Figure 3: **Color Wheel Task**

information about the stimulus (color) - setting up the orientation to be a viable "role" (role addressability) in this task.

The experimental results for this task are not discriminative between the two theories. There is evidence for both theories depending on the experiment, the task, and the assessment metric. The predictions made by each theory become harder to distinguish as the number of items increases – guessing (uniform error distribution) is hard distinguish with proper power from low precision (larger variance gaussian error distribution).

There are some experimental results that support the resource theory. For instance, there is evidence that recall precision decreases as a power law of the number of items (Smith et al., 2018). There are other experimental results that support the slots theory. For instance, there is clear evidence of guessing in the error distributions. Also reaction times plots provide some evidence for the mixture of guessing and items in memory predicted by the slots theory(Donkin et al., 2013). Considering these mixed experimental results, the two theories have also evolved to try to account for the predictions made by the other theory. For instance, the resource theory predicts higher precision when there are fewer items present. The slots theory attempts to account for this by discussing a slots + averaging model. This model states that even though there are a discrete number of slots, if there are fewer items, the same item can be represented in multiple slots. At the time of recall, those multiple representations are averaged and lead to higher recall

precision for that item (Zhang and Luck, 2008). Other propositions include hybrid models, such as the slots + resources model. This model states that there are a fixed number slots, but the amount of resource that each slot can use is flexible and can be varied (Zhang and Luck, 2008).

A new wave of studies has explored encoding methods and how the information is stored rather than the form of the limitation. This work has focused on losslessBays et al. (2009); Zhang and Luck (2008) and lossy (Nassar et al., 2018) compression.

Nassar et al. present a chunking model that supports lossy compression. As task demands increase, some items could be compressed to make space for other items. Similar items would get stored as one average representation, losing discriminative information between the two items, but opening up space for another item. This would reduce guessing but also decrease precision for the compressed items. Depending on how much and if information is compressed, experimental results could look more slots-like or resource-like, therefore unifying these ideas under a new framework. Using a theoretical model and human psychophysics, they show that information compression, "chunking" is adaptive, reflects human behavior, and can be learned using reinforcement learning (RL) (Nassar et al., 2018). One key aspect of this work is to discuss how the brain can learn when it is advantageous to chunk and connecting this with gating and navigating the credit assignment problem. The chunking mechanism and implementation is further detailed below after finishing the discussion of the factors relating to capacity.

### 0.2.2 Variable Binding and Role Addressability

Working memory is role addressable (Oberauer, 2013; Oberauer and Lin, 2017). This means that you can access an object/item from working memory using a specific attribute (the role). For example, in the color wheel task above, the orientation could be used as the role to access the rest of the information about the bar (the color). The role is learned and is either task relevant or learned through development (for example the name/face of

a person is an important attribute that could be the role).

Having a role addressable memory means that the brain has to solve the variable binding problem. Variable binding is flexibly associating the different attributes of an object together. For example, the attributes of a person includes their name, face, height, clothes, and location. Some attributes are continually changing, such as their clothes or location and need to be correctly updated with the correct object. A common error seen in working memory tasks is a misbinding error. This error is caused when an attribute is associated with the incorrect object. The frequency of such an error suggests that this is not a trivial problem for an agent to solve. The agent needs to correctly associate all of the attributes to the correct object. Further, many times one specific attribute becomes increasingly important and becomes the role attribute allowing for the role addressable recall from working memory.

### 0.2.3 Filtering Irrelevant Information

Another body of work in working memory involves understanding the underlying neural systems, specifically the prefrontal cortex (PFC) and basal ganglia (BG).

There is a vast literature studying the prefrontal cortex (PFC) during working memory tasks. Many experiments show increased and sustained activity in the PFC during working memory tasks, implying that the PFC plays an important role (Funahashi et al., 1989; Fuster and Alexander, 1971) in working memory. Further work has explored the content of the activity and its relationship to increased memory load. The relationship between memory load and sustained activity is inconclusive: some studies suggest that higher load leads to an increase in delay period activity (Linden et al., 2003; Leung et al., 2002), while others suggest no difference (Rypma and D'Esposito, 1999).

Information is maintained in PFC while the BG acts as a gating system. This is known as the frontostriatal circuit. A large emphasis in this literature is understanding how and which items are gated into and out of memory and how reinforcement learning is used to

learn those gates. This is studied through computational models (Collins and Frank, 2013; Kriete et al., 2013; O'Reilly and Frank, 2006; Frank, 2005), pharmacology (Frank and Fossella, 2011), lesion studies (Goel et al., 1997), animal electrophysiology (Schmitt et al., 2017), human imaging (Badre and D'Esposito, 2007), and human psychophysics. Empirical results suggest and modeling work predicts that BG function could impact capacity constraints and look slot-like (PFC stripes) or resource-like (distributed representations within stripes). Stripes are isolated clusters of PFC neurons. Each stripe is distinct from neighboring stripes and is thought to be under separable control by BG for input and output gating (Pucak et al., 1996; Frank et al., 2001).

There is some early investigation of how gating can impact the nature of the capacity constraint or if it can be optimized. These experiments suggest that if individuals who perform better on working memory tasks have a better ability to ignore irrelevant information (Cowan and Morey, 2006; McNab and Klingberg, 2008). Further, lesions to the BG show filtering deficits while lesions to the PFC show maintenance deficits Baier et al. (2010). These studies show how the filtering capacity is related to capacity in the context of the PFC-BG circuit and therefore motivates the usage of the PFC-BG model in this work.

### 0.2.4 Effective Capacity

There has been some work to understand these various ideas, but they have not been brought together in one unified model. In chapter 1 I propose and train a model that addresses all of these factors relating to capacity constraints.

One key concern about the terminology "capacity constraints" is that we aren't really measuring "capacity". Most of the work does not aim to count the number of neurons or neural resources. Instead, performance on working memory tasks are used as a proxy for capacity. A more apt term for this would be - "effective capacity" which better encompasses that the performance is a proxy for the function of all components of working

memory - the input gating, the maintenance and neural resources, as well as the output gating. Simply having the resources for storing the information is not enough. If one has the storage capacity for many items, but fails to store information separately or accesses the incorrect items, the working memory resources aren't used properly and cannot help performance. Therefore, effective capacity is limited by gating demands.

## 0.3 Neural Mechanisms Underlying Working Memory: Auxiliary Background

### 0.3.1 PFC Representations

The nature and organization of the PFC representations are a big subject of research. The sustained activity in the PFC could represent the original item or could be "pointers" or probes that refer back to the original item representation in parietal cortex (PC) or visual cortex (Niki and Watanabe, 1976; Takeda and Funahashi, 2002). For example, Takeda, Lara and colleagues showed that the visual location seemed to be represented in the PFC activity, implying that the representation of the objects themselves was elsewhere (Takeda and Funahashi, 2002; Lara and Wallis, 2014).

One line of work states that organization is based on stimulus-action pairs, task sets (Collins and Frank, 2013). This model suggests that PFC stores not the items, but rather the abstract latent rules that can be used in multiple contexts. For example, the PFC would hold the latent rules that govern your interaction with friends in a classroom setting versus a party setting. The full task in the work by Collins and Frank 2013 requires the model to conditionalize response to stimuli (shapes) based on the context (color). In other words, based on the color, the appropriate response to the various shapes are different. For example, as shown in Figure 4a, each context color has its own associated action sequence and based on the shape (S1 vs. S2), the model must execute the proper action sequence (i.e. for red, shape 1, correct action sequence would be +, 0, 0, 0). Context red

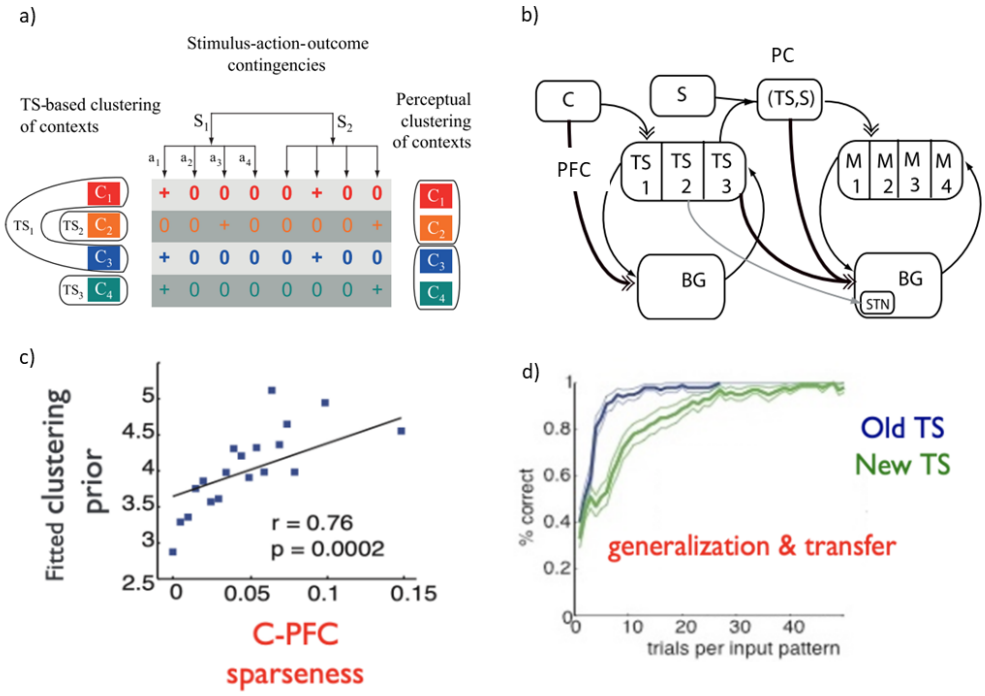



Figure 4: **Task Set Based Clustering** (Adapted from (Collins and Frank, 2013, 2016; Frank and Badre, 2012)) a) Stimuli and action pairs showing task set based clustering (clustering based on the correct action given the stimuli) on the left and perceptual based clustering (clustering based on perceptual similarity of the stimuli) on the right. b) Neural network architecture: context is fed into the PFC, which represents the task sets. Through basal ganglia gating, this narrows down the space of possibilities and is fed into the parietal cortex with the stimulus which gates the correct action. c) The connectivity between the context and the PFC layers determines the amount of clustering that is seen between contexts and task sets (example: red and blue contexts being clustered with task set 1) d) Shows the generalization and transfer that is afforded by this network.

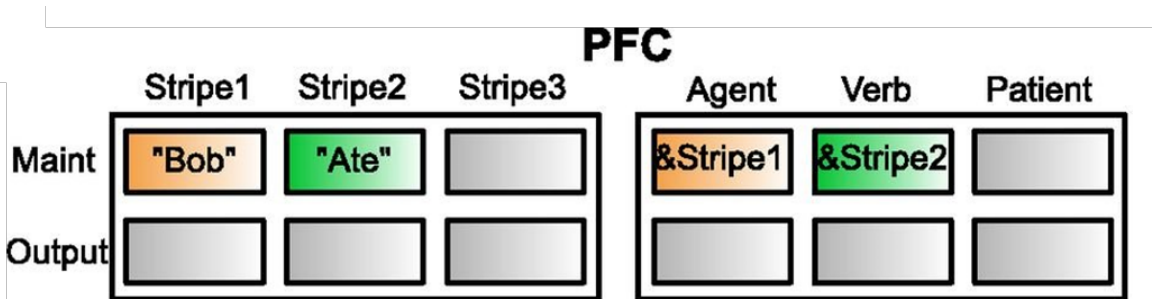



Figure 5: **Pointers Architecture** Representation in PFC that includes 3 stripes for stimuli and 3 stripes for addresses

and blue have the same action sequence and because of this, in task-based clustering, the red and blue context would be combined into one task set. The PFC would store task set 1 (red and blue stimuli), task set 2 (orange stimuli) and task set 3 (green stimuli) and with the basal ganglia, gate the correct actions, given the stimulus. If clustering were purely perceptual, red and orange would be expected to cluster together and blue and green would be expected to cluster together. The neural network architecture supported this by storing task sets in the PFC (Figure 4b), which provides the context and the posterior cortex (PC) integrates the context with the given stimulus to output the correct motor command (M). The sparsity in the connectivity matrix between context (C) and the PFC (task sets) determines the amount of clustering of contexts to the latent rules (Figure 4c). An example of clustering of contexts to latent rules is clustering the red and blue context to task set 1. This supports generalization and the authors show an advantage to task-based clustering (over no clustering) when learning a new context (Figure 4d). This result (Figure 4c) also shows that the PFC representation is indeed the task set rather than the context. Using a biologically plausible model with this architecture, this architecture can used for generalization and faster learning in the future (Collins and Frank, 2016, 2013; Frank and Badre, 2012). This architecture can be thought of in terms of the color wheel task: the orientations would be the "C" (contexts) and the colors would be the "S" (stimuli). Orientations that point to similar colors can be clustered and form a task set. Other work offers an alternate hypothesis where PFC could hold addresses or pointers in the computer science sense, that describe the type of content and location of that content (Kriete et al., 2013). In an example with language (Figure 5), the address for "Agent" is stripe 1. Using pointers within PFC that describe the type of content ("Agent") and location (stripe 1) allow for rapid generalization to unseen sentences and situations. In an example with agent, verb, and patient, PFC has stripes to hold the addresses and stripes to hold the actual content (i.e. "Bob"). This hierarchy is supported by previous work that suggests anterior areas in PFC represent more abstract information while more posterior areas represent more concrete information (Badre and D'Esposito, 2007). The

addresses and stimuli location proposed in this paper are within the PFC region, but it is possible that the stimuli could be in another region and the pointers in the PFC would work in conjunction with other regions.

There is an open discussion on where the representations live and the architecture of this memory network. While this debate is ongoing, previous modeling work (Collins and Frank, 2013; Kriete et al., 2013) has made a lot of progress in understanding how the PFC and basal ganglia can correctly gate in (filter) and gate out (read-out) using reinforcement learning (O'Reilly and Frank, 2006). The basic function that the models accomplish is to use reinforcement learning to correctly gate in, maintain, and then gate out items from working memory. These models have been corroborated to some extent by human imaging data (Badre and D'Esposito, 2007). This work gives us strong empirical and theoretical grounds for trying to use this biological system to understand capacity constraints. There is limited work in trying to understand how these biological models with a gating architecture perform under capacity constraints and perhaps probing this would lead to new insights. Since these models have gating that is learned through reinforcement learning, they may also be able to learn an optimal solution under capacity constraints that is adaptive based on situation. For example, recent work has began exploring theoretically optimal methods of encoding under capacity constraints, such as chunking, and the biologically plausible models could account for this behavior.

### 0.3.2 Chunking: A Biological Mechanism for Lossy Compression

Lossy compression, or chunking, as discussed in (Nassar et al., 2018), is defined as merging stimuli across a certain axis. Items (Fig 6a) can be described across many axes or dimensions (Fig 6b), such as color and orientation. Chunking can happen across any of these dimensions, orientation (Fig 6c) or color (Fig 6d). When two items are chunked in some dimension, both items are now represented by some combination (e.g. mean, median) of the two original stimuli in that dimension. For example, a yellow vertical bar

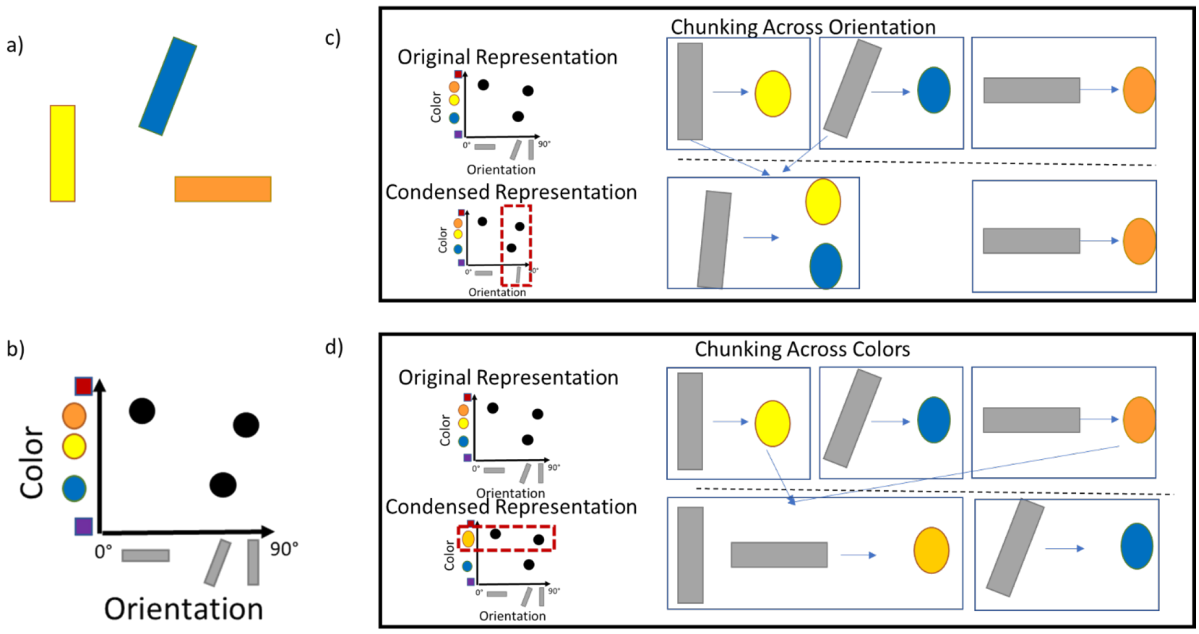



Figure 6: **Chunking** a) Original Stimuli includes 3 bars with different orientations and colors b) Items can be broken down into their composite dimensions, such as color and orientation. Here, items in (a) are mapped to these two dimensions. c) Chunking across orientations: the blue and yellow bar have similar orientation, but the colors associated with these orientations have no relationship, therefore chunking across this dimension yields loss of information for both orientations d) Chunking across colors: the bars with yellow and orange have similar color and therefore are represented by average color of darker yellow. Both the horizontal and vertical bars are now associated with this new averaged color.

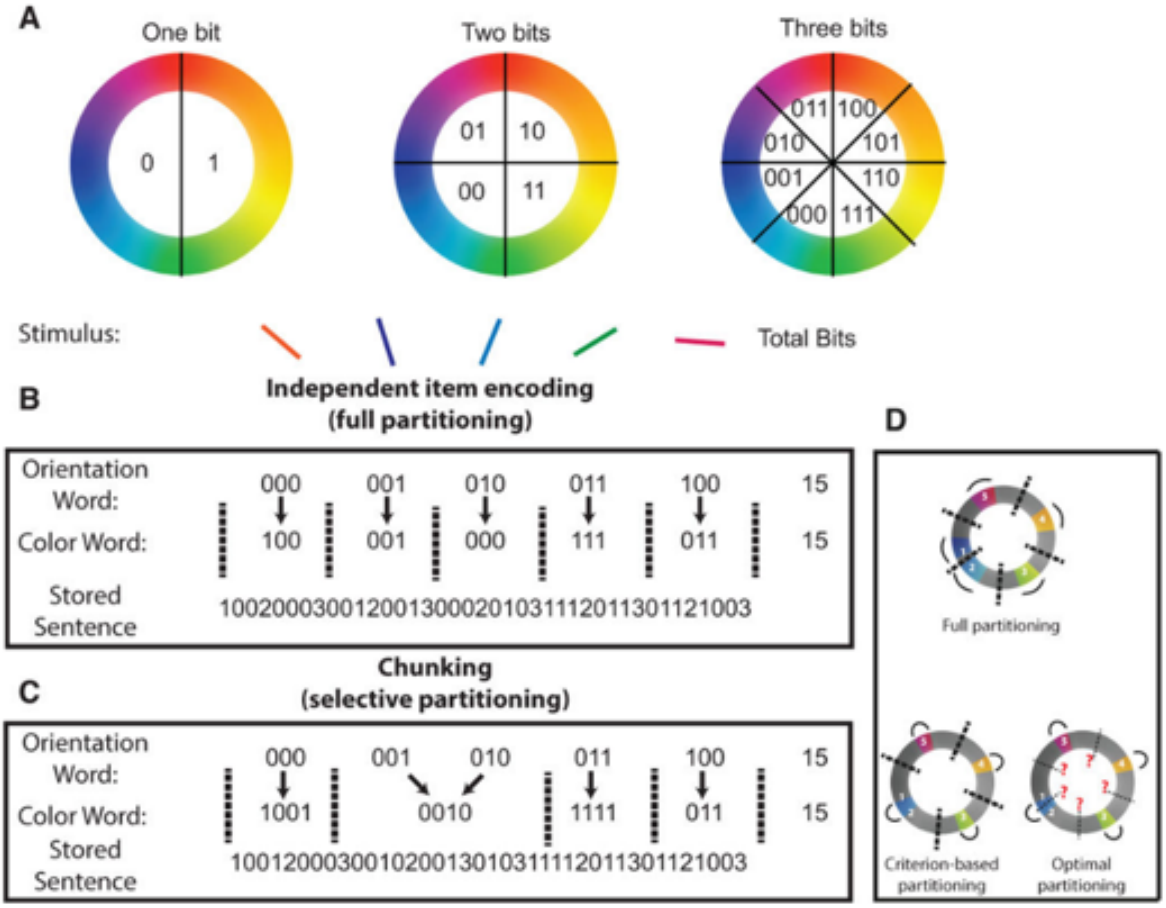



Figure 7: **Theoretical Model** (Adapted from (Nassar et al., 2018)) a) Number of bits determine storage b) No chunking example where mapping from orientation to colors is one-to-one. c) Chunking where the mapping from orientation to color is few-to-average-of-few. d) Partition types: full portioning (no chunking), criterion based portioning, and optimal partitioning.

and an orange horizontal bar could be chunked across the color axis and be represented as a horizontal and vertical bar that are both golden (Fig 6d). This frees up resources or space in memory for more items but loses precision because discrimination between the two stimuli across that dimension is no longer possible. Another way to say this, is that the mapping between colors and orientations is no longer invertible, making this a lossy compression.

In the color wheel task, orientations are used as probes – given an orientation, the participant must recall the associated color. Figure 5 has been set up with orientation

on the x-axis and the arrows pointing from the orientation to the color. Depending on the task, this could be different. Chunking across similar colors (Fig 6c), the continuous response element, may be advantageous. If there is an average statistic between the two colors that results in "accurate enough" memory for both colors, it may be advantageous to remember these two orientations mapping to that average statistic (same shade of yellow/orange) rather than the original stimuli. However, it is less advantageous to chunk across orientations. The orientations do not have a meaningful relationship with the colors, i.e. similar orientations may be associated with very different colors, chunking across orientations (Fig 6c) (remembering some average orientation for two original orientations) would lead to guessing and 50 % chance of a completely incorrect response: given the probe (i.e. horizontal bar or slightly off horizontal bar), the correct color response (blue or yellow) would be left up to guessing. Chunking across the color dimension frees up resources, however it comes at the cost of 1) not being able to distinguish the stimuli (horizontal bar and vertical bar) across the color dimension because they will both be represented by the same average color and at the cost 2) that responses to these probed orientations will be less precise because the representation will include the average color, which is some minimal distance from the original color.

Is chunking theoretically advantageous? (Nassar et al., 2018) explored this question using an abstract binary encoding model. Using 3 bits for each color and orientation (which gives $2^3 = 8$ items that can be discriminated), they can create a mapping from orientations to colors that represents various partitions. Independent encoding or no partitioning, is when each orientation maps to one color. Criterion based partitioning is where some criterion (i.e. distance of less than $\pi/8$ ) determines whether colors are chunked. Optimal partitioning is chunking that is optimal for a given situation.

To study this model, they used the color wheel task. The standard task, colors are randomly sampled independently, leading to variability across trials in how close the colors are t each other (random condition). Nassar et al. add a second condition to

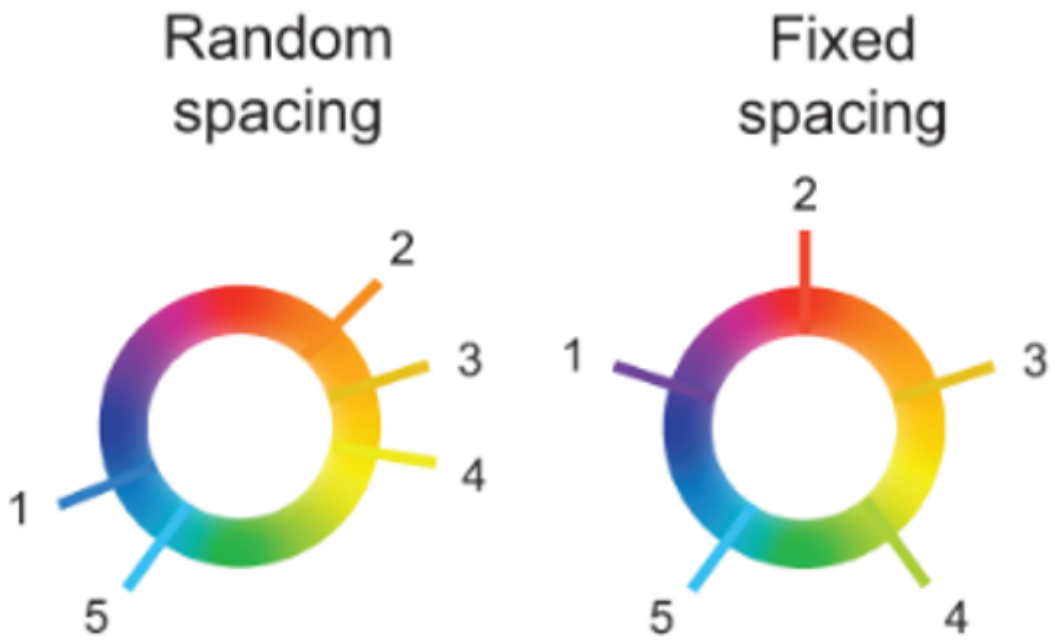



Figure 8: **Random Vs. Fixed** (Adapted from (Nassar et al., 2018)) Trials are sampled from to possible conditions. Fixed spacing ensures that items are equally apart. Random spacing allows for items to be closer or further apart.

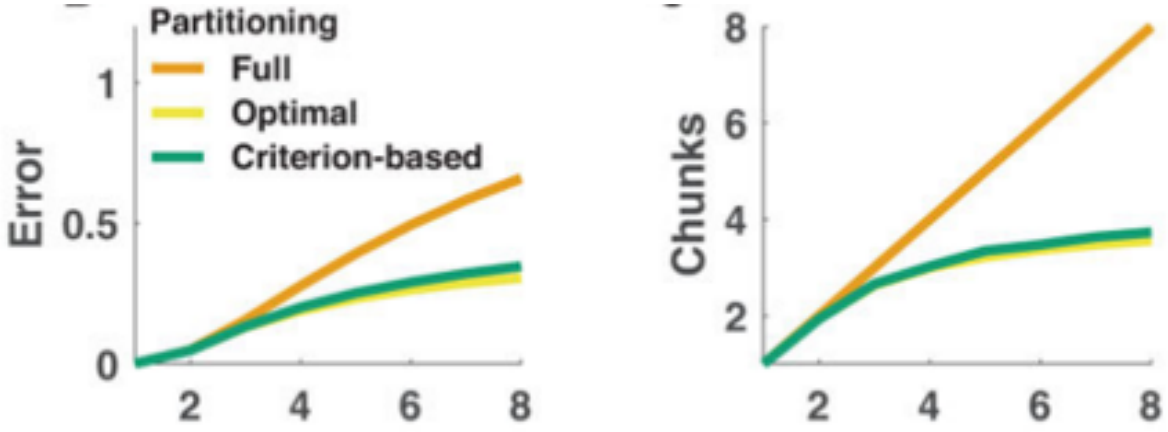



Figure 9: **Dependence on Set Size** (Adapted from (Nassar et al., 2018)) As set size increases, a)error for all partition types increases and the difference between partition and no partition also increases (the models with partitions have less error) b)and number of chunks increase for all models

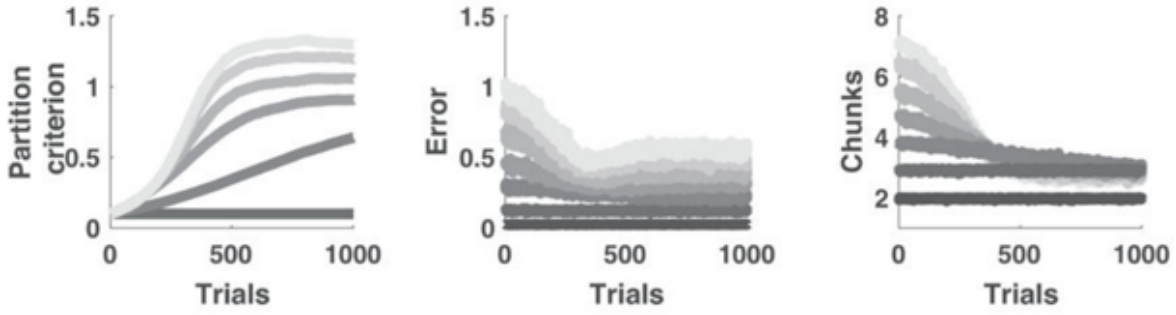



Figure 10: **Learning Through Reinforcement Learning** (Adapted from (Nassar et al., 2018)) Reinforcement learning is used to adapt chunking behavior in an advantageous manner. a) Partition criterion tends to increase over trials and increase based on set size, which leads to b) lower error and c) fewer chunks.

remove this independence and sample colors at a fixed distance between each other (fixed condition). In the fixed condition, chunking is less likely. Using this model and task, where the criterion for chunking was variable, they found a clear advantage for chunking. The partition criterion increased (i.e. chunking increased) as the number of items, set size, increased.

They also found that the model performed better on the random spacing condition, a finding that they mirrored with their human experimental data. Lastly, they found that chunking behavior (whether the model chooses to combine stimuli and decrease the number of partitions) can be modulated trial-by-trial using reinforcement learning. The partition criterion (the minimum distance needed for stimuli to be combined) was modulated by some learning rate, reward prediction error and the chunking behavior in the previous trial compared to overall chunking behavior. The authors found that the partition criterion: 1) increased from zero (zero partition criterion = no chunking), 2) was modulated trial-by-trial, 3) stabilized at different values for different set sizes, and 4) was modulated such that error was reduced. More specifically, larger set sizes lead to larger partition criterion.

In sum, chunking behavior is modulated by set size and can be effectively modulated by reinforcement learning, implying that chunking is strategic and adaptable and not a

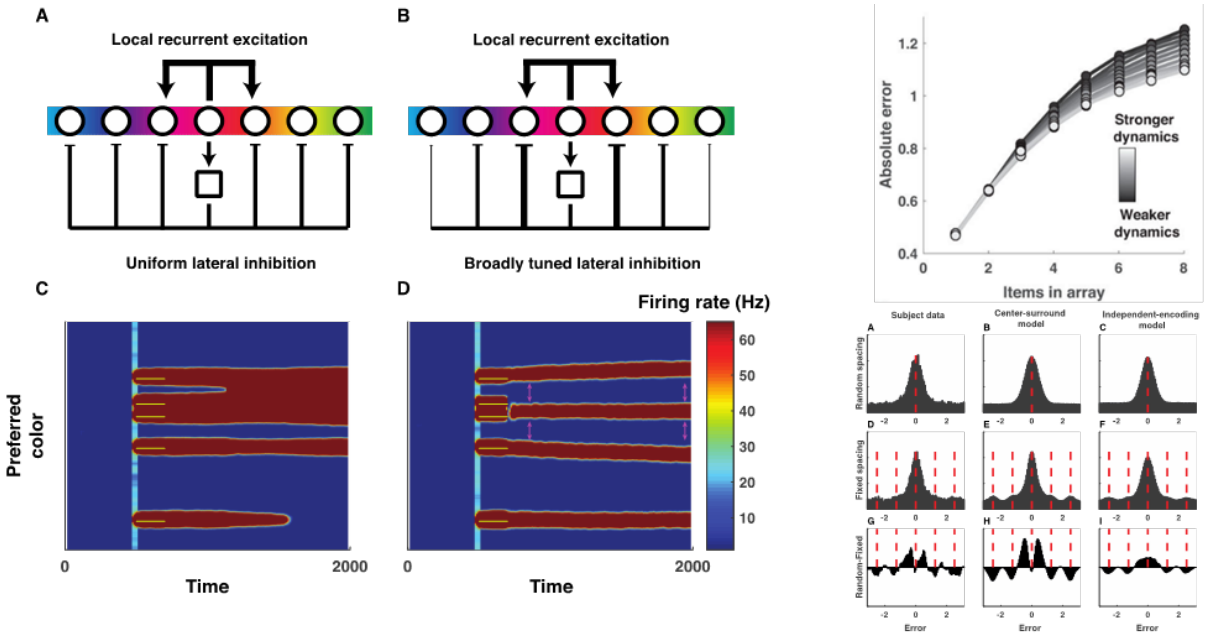



Figure 11: **Recurrent Connectivity** a) Connectivity with popular recurrent neural networks – uniform lateral inhibition b) connectivity with center surround – broadly tuned lateral inhibition. c-d) Firing across time given some color preference with the color representing the rate of firing (darker = more). c) Given a uniform lateral inhibition model, there are large chunks and forgetting of stimuli (bottom color) d) Given a center-surround mechanism, stimuli within some threshold are chunked and beyond that threshold are repelled (arrows). e) Example of increased performance given stronger center surround dynamics. f) Comparison of human data, center-surround model, and independent encoding model. The error distributions come from the random spacing condition, fixed spacing condition, and the difference between the two distributions. The distributions between the subject data matches more closely with the center surround model, as seen mostly through the difference between the random and fixed error distributions.

mere consequence of perceptual confusion.

What is a possible mechanism for chunking? Popular recurrent neural networks used to model working memory tasks (Wei et al., 2012) exhibit lateral inhibition and are able to exhibit chunking behavior (merging of two representations). Wei and colleagues used this network to create a model that integrated the slots and resources theories and exhibited properties of both. The chunking behavior seen in their model led to a decrease in precision due to large chunks (many stimuli were collapsed together), which would be predicted by the resource theory. It also led to an increase in guessing due to forgotten stimuli (due to the overwhelming effect of the large chunk), which would be predicted by the slots theory. However, their model also predicted that performance would be worse in the randomly spaced condition, which is contrary to what the theoretical binary encoding model has shown and contrary to what human experimental data has shown (Nassar et al., 2018). An alternate model that is based in biology (Somers et al., 1995; Kiyonaga and Egner, 2016) is a center surround model where the lateral inhibition is tuned rather than uniform. As explored by Nassar and colleagues, this model led to the attraction and collapse of stimuli that were within some threshold and repulsion of stimuli that were further than the given threshold. This prevented the creation of huge chunks, prevented forgetting, and overall showed an increase of performance with chunking, specifically in the random stimuli condition. As shown before with the abstract model, chunking has a theoretical advantage. Further, chunking via center-surround mechanism increases model performance and matches human experimental data better than a model with no chunking (an independent encoding model). This implementation is biologically-based, but does not address how the chunking strategy could be adapted via RL biologically. Nor does this address chunking in the full context of the working memory task, namely, does not address the complex variable binding issue. I aim to address these in concerns in chapter 1.

### 0.3.3 Oscillations

The work here focuses on computations at the cognitive and systems level. However, another body of work focuses on neural dynamic correlates of related computations. I will give a brief overview of the work on oscillations here, but it is not the focus thereafter.

An adjacent field focuses on oscillations in working memory. When electrical activity in large groups of neurons becomes coordinated, this creates measurable synchronous activity. These oscillations occur at various frequencies. Most relevant to working memory tasks is alpha, beta, theta and gamma (Roux and Uhlhaas, 2014; Lundqvist et al., 2023). Alpha oscillations are thought to help inhibit irrelevant activity - supporting the idea of suppressing gating of irrelevant items. Beta oscillations further support the role of gating into and out of working memory. Gamma oscillations support the maintenance of the working memory content. Theta oscillations temporally organize the content so it can be properly accessed. The authors highlight a spatial computing architecture involving a separation between control of working memory and the actual content of working memory, which supports a role addressable architecture such as the one proposed in this thesis. Their work also argues for a spatial structure to WM which is analogous to stripes and learning gating policies for using those stripes. Driven by transient disinhibition, PFC layers switch between attractor states (holding different item in memory). The PBWM model reflects similar ideas to oscillatory models. For example, the gating operation implementation involves rapid disinhibition of thalamocortical activity. This rapid activity could correspond to bursts of beta and theta activity. Other possible parallels exist and can be made even though it does not simulate oscillations.

## 0.4 Credit Assignment

While current work aims to understand the optimal encoding given constraints, it does not address how or why the working memory capacity constraint might emerge. Musings

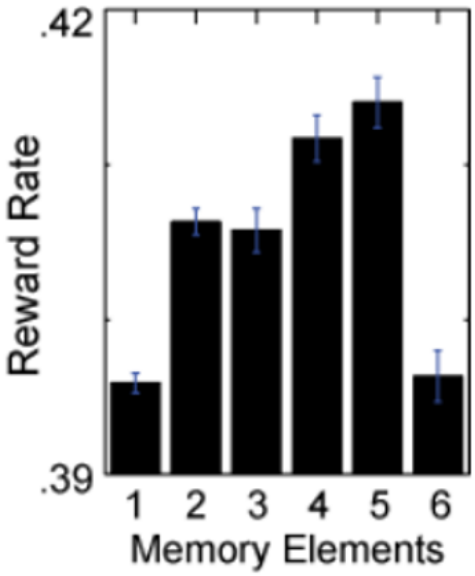



Figure 12: **Optimal Memory Elements** (Adapted from (Todd et al., 2009)) As the number of memory elements given to the model increase, the reward rate increases and then decreases, indicating there is some optimal value

from (Cowan, 2010) propose that this capacity constraint could be due to a weakness in biology or could be a strength. Mathematical simulations seem to suggest that search is optimized when the number of items is on average 3.5, suggesting a reason that the capacity constraint is actually advantageous (Cowan, 2010). Similarly, other preliminary work shows that increasing the allowed memory elements in a model does not always lead to increased performance (Todd et al., 2009). This model was an abstract prefrontal cortex – basal ganglia reinforcement learning model and here increasing the memory makes the problem of credit assignment and learning the connection between environment and reward more difficult.

These findings beg the question – is this capacity constraint due to limitations in storage or related to processing and effective manipulation of items. The question remains, which distinct process – storage of items or probing and manipulation of items – is the more likely cause for this limitation. Most current working memory models used to study capacity do not require the read-out or manipulation of items that are in memory – they only require that certain items be in the representation to be considered correct and

therefore they ignore the distinction between storage capacity and processing. One reason for this is that these models do not have a storage for the other dimension (orientation) and even if they did have storage, they would need a way to solve the binding problem between orientation and color, which is difficult. Another likely reason is that the read-out problem is difficult as it requires the model to have content addressable memory – in other words, outputs cannot be based on time, but rather must be based on external cues and this problem is much harder for a model to learn.

The credit assignment problem in working memory lies in learning the appropriate gating strategy: 1) if and where to gate information 2) when and what information to update and 3) where to read out from during recall. Correct response and reward is contingent on successfully tackling all 3 steps. Exploring and learning the correct gating operations to reveal the correct solution is referred to as the credit assignment problem.

## 0.5 Transformers

Up till now I have discussed working memory in the context of biological agents and biologically inspired neural networks. Here I describe, Transformers, which will be the main model in chapter 2, and how the ideas of working memory and credit assingment are relevant even for these large models.

The PBWM model was inspired by the Long short term memory (LSTM) neural network model (O'Reilly and Frank, 2006). The LSTM model is a popular recurrent model which includes gating components and performs well on language tasks. In multiple working memory tasks, the LSTM and PBWM training times and performance are similar. In the phonological loop task, the PBWM trains faster and makes fewer errors in generalization than the LSTM. The biological model with the ability to learn content-specific gating policies can generalize.

Following the LSTM, a newer powerful artificial neural network model class is known

as Transformers. LSTM models have now been replaced by this newer class of models. Transformers have structural components that make them reminiscent to the PBWM model. Those similarities encourage us to explore if computational principles from the PBWM model will hold in Transformers.

Transformers have no inherent capacity limit since the context window encompasses all of the information at once. The challenge for the transformer is to attend to specific items that are relevant for a given point in time and selecting what in the either context is useful. This learning problem abstractly resembles that of a gating network, therefore further validating this comparison.

Transformers are built of encoding and decoding components. The input to the Transformers is a sequence that is broken down into tokens. The tokens are fed in simultaneously. One common task is for the transformer to predict the next token in the sequence. In these tasks, masked attention is applied. Masked attention is where only previous tokens can be used to make the prediction and all subsequent tokens are hidden. The first applications for transformers were in language and translation and has now made its way into many other domains.

Transformers have a self attention mechanism that is comprised of multiple attention heads at each layer. The attention heads act as a gating system that filter the inputs to the subsequent layers. Further, the success in the Transformer model lies in the model building an understanding of how tokens relate to one another. Said differently, given a token, the model learns which other tokens are most relevant. This is similar to the working memory computations of figuring out information in the environment is relevant and the subsequent gating actions. The underlying computations of each attention head is summarized by the following (from Attention is All You Need (Vaswani et al., 2017)) :

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}}V)$$

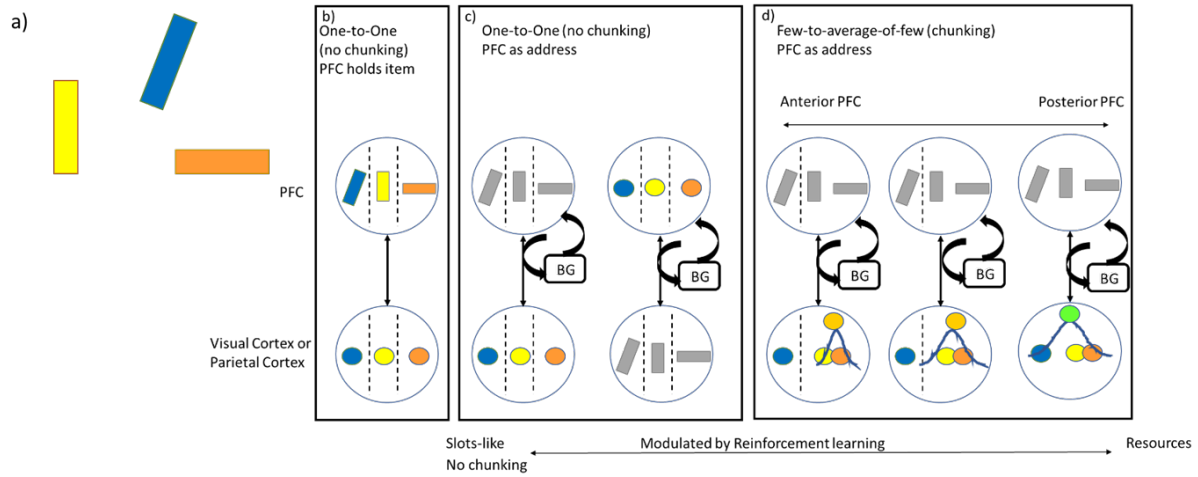



Figure 13: **Alternative Architectures** a) Stimuli b) PFC representation holds the original stimulus. c) PFC representation holds only parts of the representation (either the probe or the response element) and is connected with another area for the rest of the stimulus. There is no chunking – the mapping is one-to-one. d) PFC representation holds the pointer dimension (orientation) and projects to a different area for the response element. The response element is chunked, and the mapping is few-to-average-of-few. c-d) Reinforcement learning can modulate the amount of chunking via interactions with the BG. Further, it is possible that there are multiple representations across anterior-posterior PFC that represent different granularities that is chosen by RL.

Where Q is the query matrix, K is the key matrix, V is the value matrix, and $d_k$ is the key matrix dimension. This attention computation can be compared to input and output gating as we will show Chapter 2.

As mentioned earlier, there is no inherent capacity to the working memory of the the transformers. However, the working memory capacity limit is not the only thing governing performance. The learning problem becomes more difficult and therefore the credit assignment principle might still affect Transformer model learning and performance. In Chapter 2 we train transformers on working memory tasks that were used on humans.

## 0.6 Open Questions

While most recent work has shown that chunking is an optimal strategy to navigate capacity constraints that people seem to employ, there are many open questions that

remain. For one, current work entirely ignores the problem of read out. The brain must not only find a way to optimally store and maintain items in memory, but also manipulate and pull out the correct item given a certain probe. Existing chunking models (Nassar et al., 2018; Wei et al., 2012; Compte et al., 2000) do not solve this read out problem and therefore it is interesting to incorporate this read out problem for many reasons. One, existing biologically plausible PFC/BG neural network models do solve the read-out problem and can solve simple working memory tasks in a content addressable manner (O'Reilly and Frank, 2006). Secondly, there is debate as to where the capacity constraints are due to the storage or processing of those items. Therefore, having a model that can solve the full working memory task would be useful in probing this question. Further, as reviewed earlier, there is still debate as to the architecture of memory system and where the representations of the objects and pointers may live. One previous experiment has shown that activity in the PFC seemed to be the pointer the representation existed elsewhere (Takeda and Funahashi, 2002; Lara and Wallis, 2014). Other work suggests that the representation may be more abstract and based on actions required for given stimuli (Collins and Frank, 2013). The PFC representation might even include addresses that serve as probes or pointers and support generalization (Kriete et al., 2013). Regardless of the architecture, whether one model can solve the full WM problem is of interest. Can this WM show chunking as an adaptive strategy and can this strategy by modulated by task demands and RL? Given a model that meets these requirements, can we better understand human WM and make predictions about patient populations with WM deficits? There is a lot of room to better understand Transformer models. Do some of these computational learning principles hold in these models, despite not having a limited WM? Can we leverage potential findings to suggest training regimes for more complex Transformer tasks?

## 0.7 Contributions

In this thesis I combine the biological basis for working memory with the theoretical theories of working memory capacity. I use the PFC - BG working memory (PBWM) network developed by (O'Reilly and Frank, 2006; Moustafa et al., 2008) as the basis for this study. Through this analysis and study we showed that information compression (chunking) can be learned using reinforcement learning by the biological network. The network exhibits adaptive chunking by adjusting its learned gating policy to match task demands. The model and model errors suggest that this framework reconciles the slots and resources theories. We also find that chunking affects model learning. It suggests a provocative conclusion: simply having more WM capacity (like stripes) doesn't mean the model performs better. The real challenge lies in learning to manage those resources effectively. This is due to the credit assignment problem.

The PBWM model used was inspired by and closely compared with the artificial neural network long short-term memory (LSTM) model (O'Reilly and Frank, 2006). More recently, Transformers have surpassed the LSTM in most language tasks. In fact, their effectiveness has made them ubiquitous. Transformers are a family of artificial neural network models that make use of attention heads and parallel input processing. While powerful, Transformers end up being black boxes and a new wave of research is seeking to interpret what the models are learning - "mechanistic interpretability". In working memory tasks, gating and the fronto-striatal circuit play an important role. If trained on a working memory task, do Transformers learn in a way analogous to the frontostriatal circuit? While Transformers, do not have a working memory capacity, could the same computational principle of credit assignment apply?

In Chapter 1, I will present my manuscript (published at eLife). Here I argue how existing theories can be reconciled by the proposed framework. I also present a novel explanation for having a smaller working memory capacity. Finally, I will discuss the importance of reinforcement learning and dopamine and how this study makes important

predictions for patient populations.

In Chapter 2, I will present my manuscript (in preparation for ICLR). Here, along with my collaborators, we argue that some Transfromer models learn a gating strategy reminiscent of frontostriatal gating. The models that learn this mechanistic solution show better generalization. Finally, we show how to computational principle of credit assignment is present in Transformers and this is a learning challenge that Transformers must overcome. Our work also suggests a potential training regime where the model is pretrained on a simple task which encourages the development of a mechanistic solution.

In Chapter 3, I provide the limitations of my work, future steps, major conclusions, and discussion.

# References


Badre, D. and D'Esposito, M. (2007). Functional magnetic resonance imaging evidence for a hierarchical organization of the prefrontal cortex. *Journal of Cognitive Neuroscience*, 19.

Baier, B., Karnath, H. O., Dieterich, M., Birklein, F., Heinze, C., and Müller, N. G. (2010). Keeping memory clear and stable - the contribution of human basal ganglia and prefrontal cortex to working memory. *Journal of Neuroscience*, 30.

Bays, P. M., Catalao, R. F., and Husain, M. (2009). The precision of visual working memory is set by allocation of a shared resource. *Journal of Vision*, 9.

Bays, P. M. and Husain, M. (2008). Dynamic shifts of limited working memory resources in human vision. *Science*, 321.

Collins, A. G. and Frank, M. J. (2013). Cognitive control over learning: Creating, clustering, and generalizing task-set structure. *Psychological Review*, 120.

Collins, A. G. E. and Frank, M. J. (2016). Neural signature of hierarchically structured expectations predicts clustering and transfer of rule sets in reinforcement learning. *Cognition*, 152.

Compte, A., Brunel, N., Goldman-Rakic, P. S., and Wang, X. J. (2000). Synaptic mechanisms and network dynamics underlying spatial working memory in a cortical network model. *Cerebral Cortex*, 10.

Cowan, N. (2010). The magical mystery four: How is working memory capacity limited, and why? *Current Directions in Psychological Science*, 19.

Cowan, N. and Morey, C. C. (2006). Visual working memory depends on attentional filtering. *Trends in Cognitive Sciences*.

Donkin, C., Nosofsky, R. M., Gold, J. M., and Shiffrin, R. M. (2013). Discrete-slots models of visual working-memory response times. *Psychological Review*, 120.

Frank, M. J. (2005). Dynamic dopamine modulation in the basal ganglia: A neurocomputational account of cognitive deficits in medicated and nonmedicated parkinsonism. *Journal of Cognitive Neuroscience*, 17.

Frank, M. J. and Badre, D. (2012). Mechanisms of hierarchical reinforcement learning in corticostriatal circuits 1: Computational analysis.

Frank, M. J. and Fossella, J. A. (2011). Neurogenetics and pharmacology of learning, motivation, and cognition.

Frank, M. J., Loughry, B., and O'Reilly, R. C. (2001). Interactions between frontal cortex and basal ganglia in working memory: A computational model.

Funahashi, S., Bruce, C. J., and Goldman-Rakic, P. S. (1989). Mnemonic coding of visual space in the monkey's dorsolateral prefrontal cortex. *Journal of Neurophysiology*, 61.

Fuster, J. M. and Alexander, G. E. (1971). Neuron activity related to short-term memory. *Science*, 173.

Gilchrist, A. L., Cowan, N., and Naveh-Benjamin, M. (2008). Working memory capacity for spoken sentences decreases with adult ageing: Recall of fewer but not smaller chunks in older adults. *Memory*, 16.

Goel, V., Grafman, J., Tajik, J., Gana, S., and Danto, D. (1997). A study of the performance of patients with frontal lobe lesions in a financial planning task. *Brain*, 120.

Kiyonaga, A. and Egner, T. (2016). Center-surround inhibition in working memory. *Current Biology*, 26.

Kriete, T., Noelle, D. C., Cohen, J. D., and O'Reilly, R. C. (2013). Indirection and

symbol-like processing in the prefrontal cortex and basal ganglia. *Proceedings of the National Academy of Sciences of the United States of America*, 110.

Lara, A. H. and Wallis, J. D. (2014). Executive control processes underlying multi-item working memory. *Nature Neuroscience*, 17.

Leung, H. C., Gore, J. C., and Goldman-Rakic, P. S. (2002). Sustained mnemonic response in the human middle frontal gyrus during on-line storage of spatial memoranda. *Journal of Cognitive Neuroscience*, 14.

Linden, D. E., Bittner, R. A., Muckli, L., Waltz, J. A., Kriegeskorte, N., Goebel, R., Singer, W., and Munk, M. H. (2003). Cortical capacity constraints for visual working memory: Dissociation of fmri load effects in a fronto-parietal network. *NeuroImage*, 20.

Lundqvist, M., Brincat, S. L., Rose, J., Warden, M. R., Buschman, T. J., Miller, E. K., and Herman, P. (2023). Working memory control dynamics follow principles of spatial computing. *Nature Communications*, 14.

Ma, W. J., Husain, M., and Bays, P. M. (2014). Changing concepts of working memory.

McNab, F. and Klingberg, T. (2008). Prefrontal cortex and basal ganglia control access to working memory. *Nature Neuroscience*, 11.

Miller, G., Galanter, E., and Pribram, K. (1960). *Plans and the structure of behavior: Holt, Rinehart and Winston.*

Moustafa, A. A., Sherman, S. J., and Frank, M. J. (2008). A dopaminergic basis for working memory, learning and attentional shifting in parkinsonism. *Neuropsychologia*, 46.

Nassar, M. R., Helmers, J. C., and Frank, M. J. (2018). Chunking as a rational strategy for lossy data compression in visual working memory. *Psychological Review*, 125.

Niki, H. and Watanabe, M. (1976). Prefrontal unit activity and delayed response: Relation to cue location versus direction of response. *Brain Research*, 105.

Oberauer, K. (2013). The focus of attention in working memory—from metaphors to mechanisms. *Frontiers in Human Neuroscience*.

Oberauer, K. and Lin, H.-Y. (2017). An interference model of visual working memory. *Psychological Review*.

O'Reilly, R. C. and Frank, M. J. (2006). Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Computation*, 18.

Pucak, M. L., Levitt, J. B., Lund, J. S., and Lewis, D. A. (1996). Patterns of intrinsic and associational circuitry in monkey prefrontal cortex. *Journal of Comparative Neurology*, 376.

Roux, F. and Uhlhaas, P. J. (2014). Working memory and neural oscillations: - versus - codes for distinct wm information? *Trends Cogn Sci*, 18.

Rypma, B. and D'Esposito, M. (1999). The roles of prefrontal brain regions in components of working memory: Effects of memory load and individual differences. *Proceedings of the National Academy of Sciences of the United States of America*, 96.

Schmitt, L. I., Wimmer, R. D., Nakajima, M., Happ, M., Mofakham, S., and Halassa, M. M. (2017). Thalamic amplification of cortical connectivity sustains attentional control. *Nature*, 545.

Smith, P. L., Corbett, E. A., Lilburn, S. D., and Kyllingsbaek, S. (2018). The power law of visual working memory characterizes attention engagement. *Psychological Review*, 125.

Somers, D. C., Nelson, S. B., and Sur, M. (1995). An emergent model of orientation selectivity in cat visual cortical simple cells. *Journal of Neuroscience*, 15.

Takeda, K. and Funahashi, S. (2002). Prefrontal task-related activity representing visual cue location or saccade direction in spatial working memory tasks. *Journal of Neurophysiology*, 87.

Todd, M. T., Niv, Y., and Cohen, J. D. (2009). Learning to use working memory in partially observable environments through dopaminergic reinforcement. In *Advances in Neural Information Processing Systems 21 - Proceedings of the 2008 Conference*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wei, Z., Wang, X. J., and Wang, D. H. (2012). From distributed resources to limited slots in multiple-item working memory: A spiking network model with normalization. *Journal of Neuroscience*, 32.

Zhang, W. and Luck, S. J. (2008). Discrete fixed-resolution representations in visual working memory. *Nature*, 453.

# CHAPTER 1

# Adaptive chunking improves effective working memory capacity in a prefrontal cortex and basal ganglia circuit

Aneri Soni and Michael Frank

Published in *eLife*, 2024

## 1.1 Abstract

How and why is working memory (WM) capacity limited? Traditional cognitive accounts focus either on limitations on the number or items that can be stored (slots models), or loss of precision with increasing load (resource models). Here we show that a neural network model of prefrontal cortex and basal ganglia can learn to reuse the same prefrontal populations to store multiple items, leading to resource-like constraints within a slot-like system, and inducing a trade-off between quantity and precision of information. Such "chunking" strategies are adapted as a function of reinforcement learning and WM task demands, mimicking human performance and normative models. Moreover, adaptive performance requires a dynamic range of dopaminergic signals to adjust striatal gating policies, providing a new interpretation of WM difficulties in patient populations such as Parkinson's disease, ADHD and schizophrenia. These simulations also suggest a computational rather than anatomical limit to WM capacity.

# Introduction

It has long been appreciated that working memory (WM) capacity is limited, but despite many decades of research, the nature of these limitations remains controversial. For example, early work by Miller (1951) famously posited that WM is limited to roughly 7 items, but he also stated that the precise limitation might vary depending on information quantity of the relevant memoranda. More recently, a vigorous debate over the last two decades has divided roughly into two schools of thought. The "slots" theory argues that WM is limited to a specific number of items. According to this account, each item is stored in a discrete slot and encoded with high precision, leading to low or zero errors when that item is probed at recall. When the number of items to be remembered exceeds the capacity (roughly 4 items; Cowan (2008)), some items will be forgotten, and therefore participants resort to guessing (Zhang and Luck, 2008; Fukuda et al., 2010; Luck and

Vogel, 2013). In contrast, the "resource" theory argues that people can store an arbitrarily large number of items in WM, with no inherent item limit, but that each item competes for a shared pool of resources. As a result, the precision of each memoranda goes down with each added item to be recalled. In the limit, when many items are presented, each one is recalled with low precision, which can masquerade as guessing (Bays et al., 2009; Ma et al., 2014).

Critically, regardless of the distinction between discrete and continuous resources, the measured WM "capacity" from experimental data is not fixed. For example, individual differences in WM capacity are largely determined not by the raw number of items one can store but rather one's ability to filter out distracting items (Vogel et al., 2005; McNab and Klingberg, 2008; Astle et al., 2014; Feldmann-Wüstefeld and Vogel, 2019). More generally, one can leverage various (potentially unconscious) memory strategies to improve "effective capacity", leading to experimentally observed capacity measurements that fluctuate depending on stimulus complexity, sensory modality, and experience with the stimuli (Pusch et al., 2023). Thus a key but often overlooked aspect of WM lies in the efficient *management* of access to and from working memory. Taking into account this management may also provide a mechanism for understanding WM not only in terms of maintenance, but also how gating strategies may be used for manipulation of information – i.e., "working with memory" (Moscovitch and Winocur, 2009). In other words, **effective capacity** encompasses gating items into/out of WM as well as the maintenance of items.

For example, recent theoretical and empirical work suggested that information stored in WM can be partitioned into discrete representations, but that similar items could be stored in a shared partition, in effect chunking them together (Nassar et al., 2018). Intuitively, it is simpler to remember that one needs to purchase dairy items, bread, and fruit rather than to remember to buy milk, cheese, bread, oranges, and bananas. This active chunking strategy serves as a lossy information compression mechanism: it frees up space for other items to be stored and recalled. This happens at the cost of reducing

precision for the chunked items. Experimental evidence provided support for such a model over alternatives, and provided a mechanism to explain why precision can be variable across trials as posited by previous resources models (Berg et al., 2012). Moreover, (Nassar et al., 2018) showed that the optimal chunking criterion (i.e., how similar two items need to be to merit storing as a single chunk) varies systematically with set size (number of items to be remembered) and can be acquired via reinforcement learning (RL). In line with this account, they reported evidence that humans adapted chunking on a trial by trial basis as a function of reward feedback in their experiment. They also performed a meta-analysis of other visual working memory (VWM) datasets showed that optimal performance was associated with more chunking with increasing set size. Thus, at the cost of small errors (due to loss in precision), normative models and humans have overall better recall and performance when employing this chunking method. This and related theories (Brady et al., 2011; Brady and Alvarez, 2015; van den Berg et al., 2014; Wei et al., 2012; Swan and Wyble, 2014; Nassar et al., 2018) may reconcile differences between the slots and resources theories.

While these normative and algorithmic models are consistent with experimental data, it is unclear how *biological* neural networks could perform such adaptive and flexible chunking. If a biological neural network exhibits the same mechanism, we can make more clear predictions about human WM as well. The dominant neural model of visual working memory is the ring attractor model of prefrontal cortex (PFC), whereby multiple items can be maintained via persistent activity in attractor states (Edin et al., 2009; Wei et al., 2012; Nassar et al., 2018). In these models, nearby attractors coding for overlapping visual stimuli can collide, leading to a form of chunking and loss of precision, and where some items are forgotten due to lateral inhibition (Wei et al., 2012; Almeida et al., 2015; Nassar et al., 2018). While these models have successfully accounted for a range of data, by modeling only the PFC (or a single cortical population), they have limitations. Firstly, these models cannot determine whether or not an item should be stored. In other words,

unlike humans (Vogel et al., 2005; McNab and Klingberg, 2008), they cannot improve effective capacity by filtering content to only include relevant information. Secondly, any chunking that occurs in these models is obligatory – determined only by how overlapping the neural populations are and hence whether attractors will collide. Thus, chunking can't be adapted with task demands as required by normative models and human data (Nassar et al., 2018). Finally, during recall, these network models cannot select a specific item from memory based on a probe (accuracy in these models is considered high as long as the relevant stimulus is encoded somewhere in the pool of neurons; (Edin et al., 2009; Wei et al., 2012; Almeida et al., 2015)). In other words, these models have no way of manipulating or accessing the contents of the WM.

This selective access and management of information in WM requires the brain to 1) solve the variable binding problem and 2) create a role- addressable memory. **Variable binding** refers to the ability to link attributes of an object together (for instance a person has a name, face, location etc.) In WM, humans have to temporarily bind a given memorandum with a given slot in memory, often in a role-addressable manner. For example, they might need to recall the color of an object based on its shape. Indeed, limitations in WM capacity have been linked to difficulty in binding items to their respective slots in memory(Oberauer, 2013; Oberauer and Lin, 2017). Moreover, the selective updating of information in these slots is thought to produce a recency bias ubiquitously observed in human WM (Oberauer et al., 2012).

Another complementary line of biologically-inspired neural network models addresses how interactions between basal ganglia (BG), thalamus, and PFC support independent updating of separable PFC "stripes" (anatomical clusters of interconnected neurons that are isolated from other stripes; (Levitt et al., 1993; Pucak et al., 1996; Frank et al., 2001). These prefrontal cortex basal ganglia working memory (PBWM) models focus on the aforementioned "management" and variable binding problem. They simulate the brain's decision whether to encode a sensory item in WM ("selective input gating"). They also

simulate which item (of those stored in WM) should be accessed ("output gating") for reporting or subsequent processing (O'Reilly and Frank, 2006; Hazy et al., 2007; Krueger and Dayan, 2009; Stocco et al., 2010; Frank and Badre, 2012; Kriete et al., 2013). The combination of input and output gating decisions that are made can be summarized as the *g*ating policy. Via dopaminergic reinforcement learning signaling in the BG, the networks learn an effective gating policy for a given WM task (Frank and Badre, 2012). This policy includes (i) whether or not to store an item (i.e., if it is task-relevant or distracting), (ii) if relevant, in which population of PFC neurons to store it, and (iii) which population of PFC neurons should be gated out during recall or action selection. As such, PBWM networks can perform complex tasks that require keeping track of sequences of events across multiple trials while also ignoring distractors. The PBWM framework also accords with multiple lines of empirical evidence, ranging from neuroimaging to manipulation studies, suggesting that the BG contributes to filtering (input gating) of WM (which improves effective capacity) (McNab and Klingberg, 2008; Cools et al., 2007, 2010; Baier et al., 2010; Nyberg and Eriksson, 2016) and selecting among items held in WM (output gating; (Chatham et al., 2014)). Evidence also supports the PBWM prediction that striatal DA alters WM gating policies analogous to its impact on motor RL (O'Reilly and Frank, 2006; Moustafa et al., 2008); for review see (Frank and Fossella, 2011). Finally, these human studies are complemented by causal manipulations in rodent models implicating both striatum and thalamus as needed to support WM maintenance, gating and switching (Rikhye et al., 2018; Nakajima et al., 2019; Wilhelm et al., 2023). However, to date, these PBWM models have only been applied to WM tasks with discrete stimuli and thus have not addressed the tradeoff between precision and recall in VWM. Due to the discrete nature of the stimuli, accuracy is typically binary, and WM information could not be adaptively chunked. Further, previous PBWM studies only trained and tested within the allocated capacity of the model, limiting the application to common human situations in which set size of relevant items goes beyond WM capacity.

In sum, these two classes of neural WM models address complementary phenomena but their intersection has not been studied. In particular, how can our understanding of BG-PFC gating inform the slots vs resources debate and the nature of WM capacity limits more generally? On the surface, PBWM is a slots model: it has multiple, isolated PFC "stripes" that can be independently updated and accessed for read-out. Note, however, that performance is improved in these models when they use distributed representations within the stripes (Hazy et al., 2007; Kriete et al., 2013), which can have resource constraints (Frank and Claus, 2006). We thus considered whether PBWM could acquire, via reinforcement learning, a gating strategy whereby it stores a "chunked" representation of multiple items within the same stripe, leaving room for other stripes to store other information and *in effect increasing the effective capacity without increasing the allocated capacity*.

Here, we sought to combine successful aspects of both models. We considered whether PBWM could be adapted to perform VWM tasks with continuous stimuli and whether it can learn a gating policy via RL that would support chunking to meet task demands. We include a ring attractor model that allows for mergers of continuous-valued stimuli via overlapping representations (Wei et al., 2012; Edin et al., 2009; Almeida et al., 2015; Nassar et al., 2018). But rather than representing all input stimuli at once, the network evaluates a single sensory input in the context of stimuli already stored in one or more PFC stripes. The ring attractor can then merge or chunk the sensory input with its nearest neighbor in PFC. Importantly, the resulting chunks are not obligatorily stored in WM. Rather, the BG learns a gating policy so that it can potentially store the original (and more precise) sensory input, or it can replace a currently stored representation with the chunk – and adaptively alter its strategy as a function of task demands (set size). During recall, the network can gate out the corresponding (original or chunked) representation linked to the probe, and reproduce hallmarks of human performance in continuous report VWM tasks.

Notably, we find that this chunk-augmented PBWM network outperforms control models that lack chunking abilities across a range of task conditions. Chunk models outperform control networks even when the control models are endowed with an allocated capacity that exceeds set size. This latter result stems from a credit assignment problem that arises when networks must learn to store and access multiple items in WM. Chunking instead allows for a common set of stripes to be repeatedly reused and reinforced, limiting the number of possible solutions explored. As such, this result lends insight into a normative rationale for why WM capacity is limited in the first place. Moreover, these performance advantages depend on a healthy balance of BG dopamine signaling needed to support adaptive gating policies that enhance effective capacity, providing a novel account for WM deficits resulting from aberrant BG DA signaling in patient populations such as Parkinson's disease and schizophrenia (Frank, 2005; Moustafa et al., 2008; Cools, 2006; Cools et al., 2010; Maia and Frank, 2017). Finally, we show that, like humans, the model shows a recency bias, with increased accuracy for reporting items that had been presented more recently. This results both from an increased propensity to update items that had been presented earlier and as a consequence of chunking of earlier items. This account is consistent with evidence in the human literature on the nature of recency effects Oberauer et al. (2012), and inconsistent with alternative neural models of passive decay.

# Method

The model is implemented using an updated version of the Leabra framework (O'Reilly et al., 2024), written in the Go programming language (see https://github.com/emer/emergent). All of the computational models, and the code to perform the analysis, are available and will be published on our github account. We first outline the basic neuronal framework before elaborating on the PBWM implementation, modifications to the continuous report task, and chunking implementation, with most details in the Supplementary Materials.

Leabra uses point neurons with excitatory, inhibitory, and leak conductances contribut-

ing to an integrated membrane potential, which is then thresholded and transformed to produce a rate code output communicated to other units.

The membrane potential $V_m$ is updated as a function of ionic conductances $g$ with reversal (driving) potentials $E$ according to the following differential equation:

$$\begin{aligned} C_m \frac{dV_m}{dt} &= g_e(t)\bar{g_e}(E_e - V_m) + \\ & \quad g_i(t)\bar{g_i}(E_i - V_m) + \\ & \quad g_l(t)\bar{g_l}(E_l - V_m), \end{aligned} \tag{1.1}$$

where $C_m$ is the membrane capacitance and determines the time constant with which the voltage can change, and subscripts $e$, $l$ and $i$ refer to excitatory, leak, and inhibitory channels respectively.

The excitatory net input/conductance $g_e(t)$ is computed as the proportion of open excitatory channels as a function of sending activations times the weight values:

$$g_e(t) = \langle x_i * wi \rangle = \frac{1}{n} \sum_i x_i w_i \tag{1.2}$$

Activation communicated to other cells ($y_j$) is a thresholded ($\Theta$) sigmoidal function of the membrane potential with gain parameter $\gamma$:

$$y = \frac{1}{\left(1 + \frac{1}{\gamma[g_e - g_e^{\Theta}]_+}\right)} \tag{1.3}$$

where $g_e^{\Theta}$ is the level of excitatory input conductance that would put the equilibrium membrane potential right at the firing threshold $\Theta$ and depends on the level of inhibition and leak.

$$g_e^{\Theta} = \frac{g_i(E_i - \Theta) + g_l(E_l - \Theta)}{\Theta - E_e} \tag{1.4}$$

Further details are in the Supplementary Materials, but we elaborate the inhibition function below given its relevance for the chunking mechanism.

## Base PBWM model

The PBWM model is built based on a large repertoire of research surrounding WM, gating, and RL and has been developed over a series of articles (see introduction). Here we focus on the high level description of its functionality and the unique additions to the current application, particularly the implementation of continuous rather than discrete representations throughout the network (input, PFC and response layers), and the chunking mechanism.

We modified PBWM to accommodate continuous representations such as those used in the delayed report color wheel task (Berg et al., 2012; Nassar et al., 2018), see Figure 1.2a, a common task to assess precision and recall characteristics of VWM and which forms the main basis for our simulations. In this task, participants are presented with multiple colored bars on a visual display where each bar has two attributes: a color and orientation. After some delay, participants are shown only one of the orientations in gray, and the subject's task is to report the color that was associated with that orientation using a continuous color wheel. Previous PBWM applications used only discrete stimuli and did not address precision. To simulate continuous report tasks, we represented the color for each stimulus as randomly varying from 0 to 2pi using a population code, where each neuron in the layer maximally responds for a particular color, and the full representation for a given continuous input is represented as a Gaussian bump over an average of 10 neurons in a 20 neuron layer. This representation accords with that seen in visual area V2, with hue neurons that are spatially organized according to color (Xiao et al., 2003). Each color was presented to the network together with a separate representation of its associated

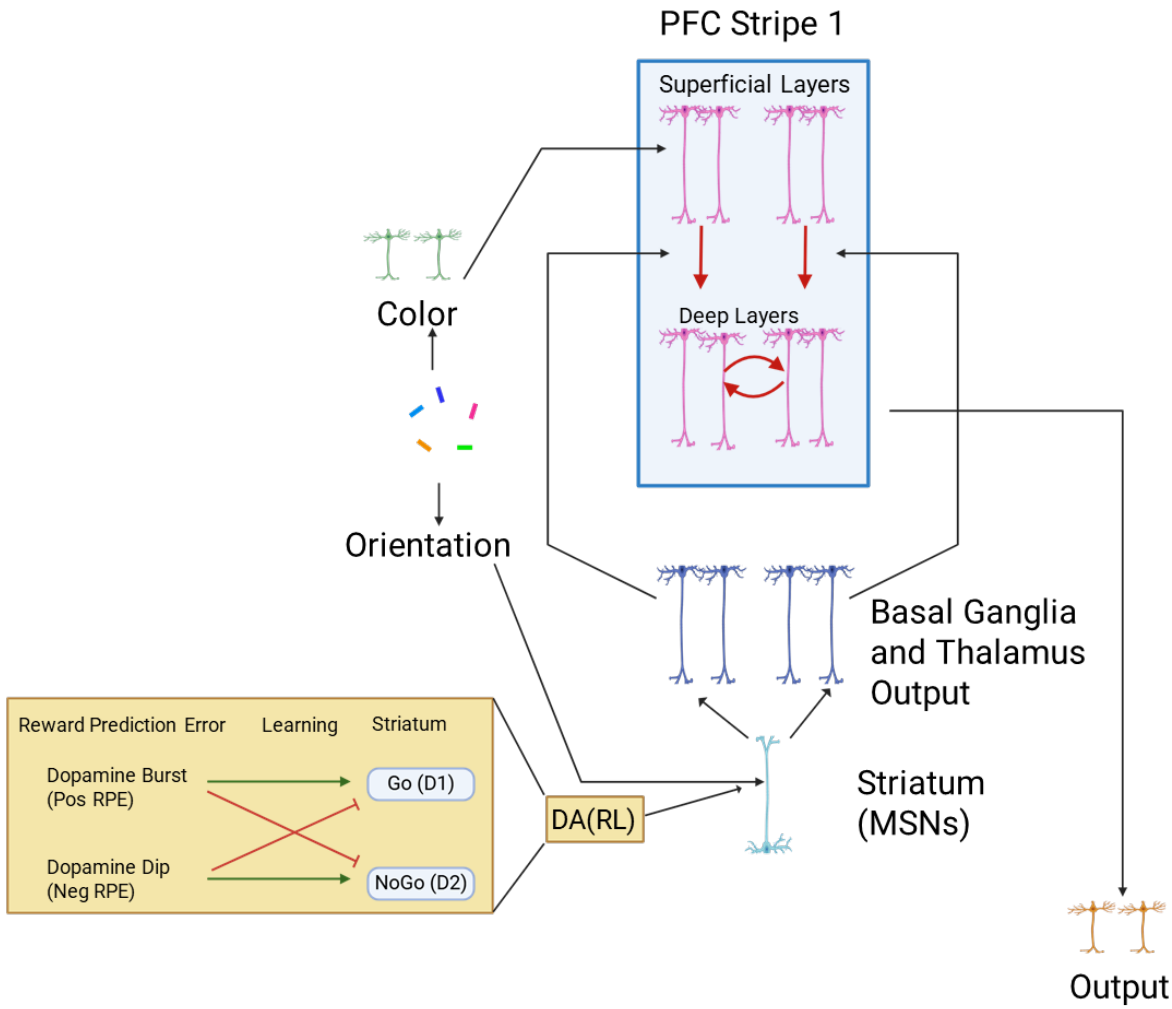



Figure 1.1: **Base Model** Sensory inputs (reflecting visual cortical representations) project to PFC superficial layers, which transiently represent those inputs. Activity is maintained in PFC after stimulus offset (and into subsequent trials) only when gated. Red arrows indicate gating, supporting transfer of information from superficial layers to deep layers, triggered by striatal disinhibition of dorsomedial thalamocortical activity. Maintenance is represented by recursive red arrows in deep layer. Green insert shows how striatum D1 and D2 neural populations (which have opposite effects on gating) are modulated by dopaminergic reward prediction errors (RPEs). Over the course of learning, synaptic weights evolve to support effective gating strategies that increase rewards.

orientation (for simplicity we used discrete orientations, as the task is not to recall the precise orientation but only to use it to probe the color). The stimuli are presented sequentially to the mode. This serves 3 purposes: to simplify the binding problem, to mimic a sequential attentional mechanism, and to make contact with literature on serial WM tasks.[1].

These input layers project to the PFC maintenance layers, which contain isolated populations in discrete stripes, also coded as Gaussian bumps of activity. As in prior PBWM models, the PFC is divided into superficial and deep layers (O'Reilly and Frank, 2006; Hazy et al., 2007; Frank and Badre, 2012; Hazy et al., 2021). The superficial PFC layers for each stripe will always reflect the input stimuli transiently, as candidates to be considered for storage in WM. But for these stimuli to be maintained robustly over delays (and over intervening other stimuli on subsequent time points), they have to be gated into WM. Accordingly, each PFC stripe is modulated by a corresponding BG module consisting of striatal "Go" and "NoGo" units which in turn, via direct and indirect pathways, project to BG output / thalamic units. When there is relatively more Go than NoGo activity in a given module, the corresponding Thalamic output unit is activated, inducing thalamocortical reverberation and activation of intrinsic ionic maintenance currents, thereby triggering robust maintenance of information in the deep PFC maintenance layers (O'Reilly and Frank, 2006; Hazy et al., 2007; Frank and Badre, 2012; Hazy et al., 2021). Thus only the stripes that have been input gated continue to maintain the most recent color representation in an attractor over time. Importantly, gating in PBWM implements a form of "role-addressable" memory: the decisions about whether and which stripe to gate colors into depends on its assigned role. In this case, the orientation probe associated with the color determines where it should be gated. By

[1]While this sequential presentation of stimuli simplifies the binding problem, it also adds a further challenge as the network must have capacity to recall items that were presented several time steps/trials ago. To solve this problem, the model must learn an efficient and appropriate gating strategy. This also makes the model vulnerable to serial dependence it its chunking, consistent with that observed empirically, whereby WM reports are biased towards the last stimulus that was presented Bliss et al. (2017); Kiyonaga et al. (2017); Fischer and Whitney (2014)

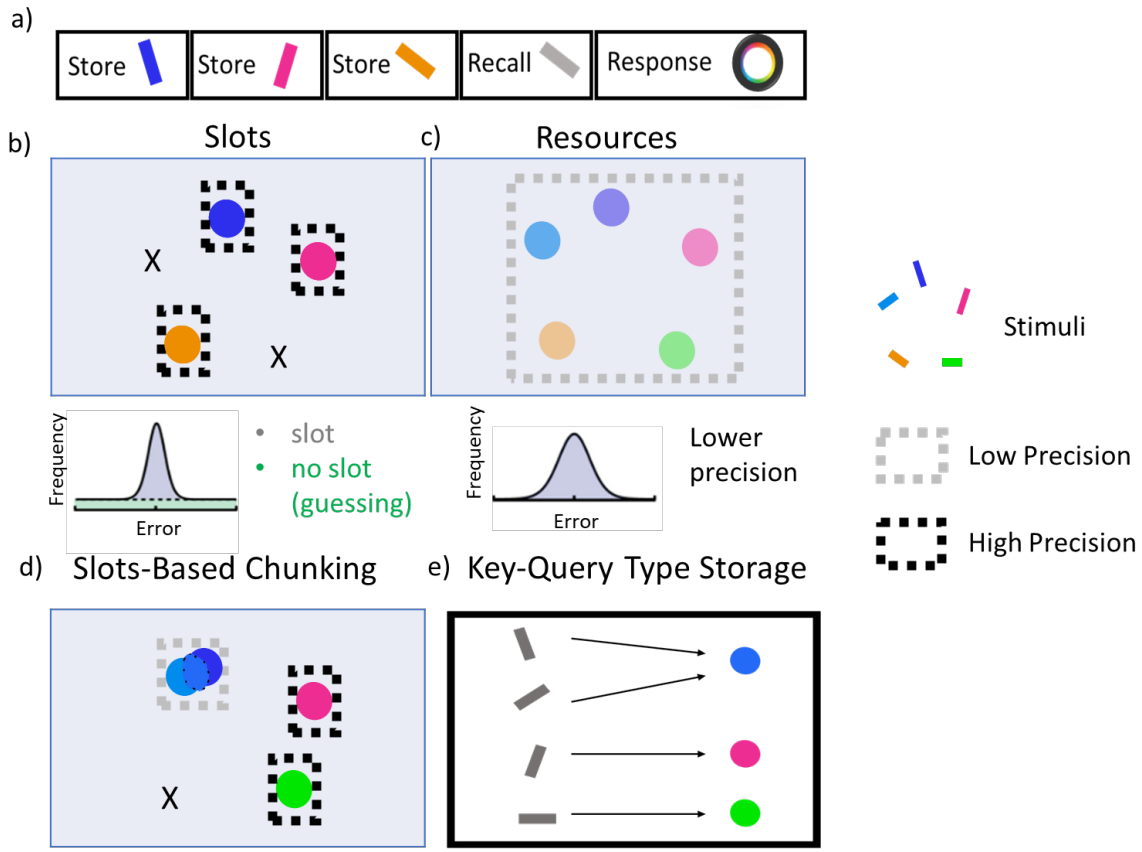



Figure 1.2: **Visual Working Memory Task.** a) The color wheel task is commonly used to study the nature of capacity limitations in VWM. During encoding, participants are presented multiple randomly generated oriented and colored bars. After a delay they are shown a recall probe trial in which one of the previously seen orientations is presented in gray. The participant responds by using a color wheel in an attempt to reproduce the color associated with that probe orientation. The number of store items are dictated by **set size**. b) Slots models suggest that WM capacity is limited by a fixed number of slots. When set size exceeds capacity, some items are stored in memory with high precision while the rest are forgotten, resulting in an error histogram that is a mixture of high precision memory (for items in a slot) and guessing (for items not in a slot). c) Resource models state that all items can be stored in a common pool, but as the number of items increase, the precision of each representation decreases, resulting in an error histogram with a large variance (but no guessing). Adapted from (Ma et al., 2014). d) A hybrid chunking model containing discrete slots, but with resource-like constraints within each slot. Here, the two bluish items are merged together within a slot, reducing their precision but freeing up other slots to represent pink and green items with high precision. The orange item is forgotten. The criterion for chunking can be adapted such that error histograms will look more like the slots theory or resource theory depending on task demands (WM load and chunkability of the stimulus array; (Nassar et al., 2018)). e) Storage in the PBWM-chunk model is like a key-query. The colors are stored as continuous representations in PFC and can be merged. The orientations are the queries used to probe where information should be stored and where to read it out from.

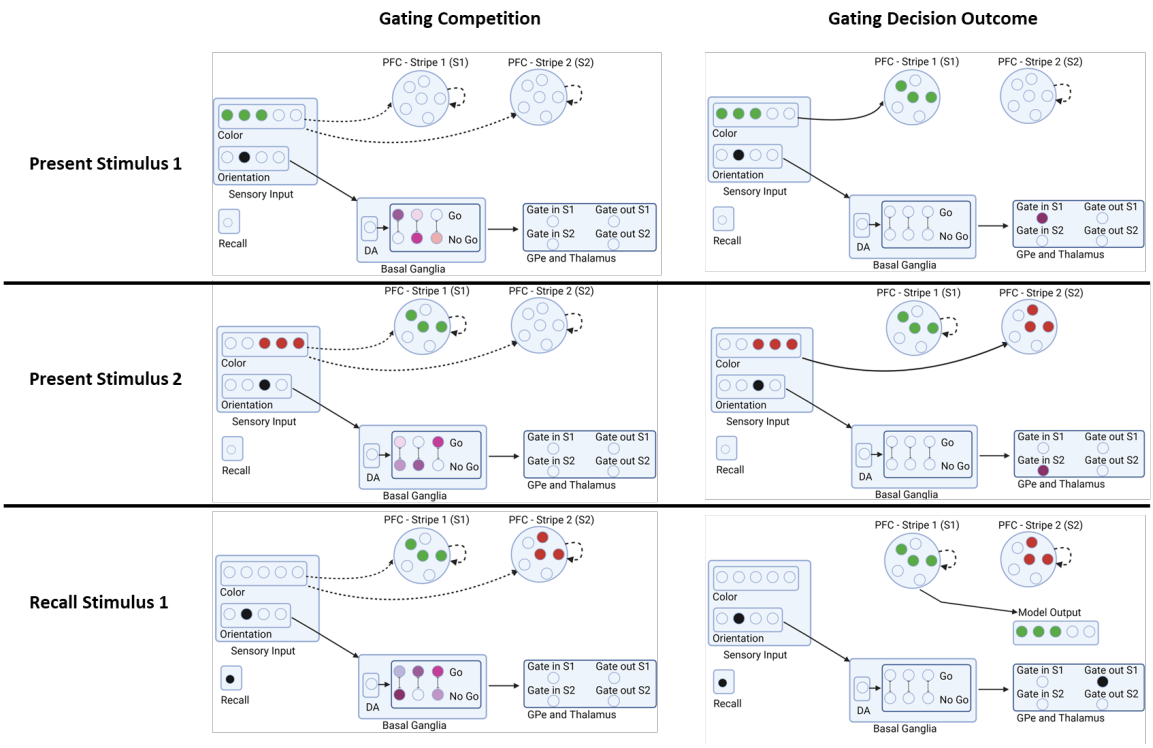



Figure 1.3: **Example Sequence of Network Gating Decisions.** In this example trial, the network is presented with stimulus 1 (color and orientation), stimulus 2, and is then asked to recall the color of stimulus 1 based on just its orientation. Each step is broken into a gating competition that involves the Basal Ganglia (striatal Go/NoGo units, Gpe/Gpi) and Thalamus units. The outcome of this internal competition determines the gating decision and the model output. When the first stimulus is presented, the relative activities determine if and where the stimulus is gated in (stripe 1 or stripe 2). The network gates stimulus 2 in a different stripe based on its orientation. During recall, the network uses a gating policy to output gate the stripe corresponding to the probed orientation. A reward is delivered to the network proportional to the accuracy in reporting the original color. A negative reward is delivered if the color is not sufficiently close (see Methods). Rewards translate into dopaminergic reward prediction error signals that serve to reinforce or punish recent gating operations. This schematic is illustrative; the actual network contains a broader population code and the PFC stripes are divided into input and output representations, each with deep and superficial layers (see Text).

receiving inputs from the orientations, the BG can thus learn a gating policy whereby it consistently stores some orientations into a particular PFC stripe, making it accessible for read out. Figure 1.3 shows a schematic example in which, based on the orientation the network gates the first PFC stripe to store the green color, but then stores the color of the second orientation to store the red color. Thus, the PBWM stripes serve a variable binding function (O'Reilly and Frank, 2006) which can also be linked to key/query coding

(Traylor et al., 2024; Swan and Wyble, 2014): in this case the network can learn to use the orientations to guide which stripe is accessed, and the population within the stripe represents the content (color).

During a recall trial, the network is presented with only a single orientation probe. The model needs to correctly "output gate": select from multiple PFC maintenance stripes, so that only a single representation is activated in the corresponding PFC "output stripes", which in turn projects to the output layer (Figure 1.3 bottom row). If it correctly recalls the associated color (by activating the color population in the output layer) it receives a reward. Rewards were given in a continuously linear fashion based on the difference between output and the target color. The activity of the output neurons was decoded (using a weighted linear combination of neuron activities; (Almeida et al., 2015)) and the reward given was inversely proportional with the error. Correctly recalling a color thus requires not only having it stored in PFC, but reading out from the correct stripe that corresponds to the probed item. This read-out operation involves a BG output gating function (Hazy et al., 2007; Frank and Badre, 2012; Kriete et al., 2013), facilitating transmission of information from a PFC maintenance stripe to the corresponding PFC output stripe. In this way, the model can read out from its several stored WM representations according to the current task demands (see (Chatham et al., 2014) for neuroimaging evidence of this BG output gating function). The input and output gating mechanism in PBWM performs a role-addressable gating function that can be linked to the key-query operations in Transformers ((Traylor et al., 2024)).

Note that successful performance in this and other WM tasks (Hazy et al., 2007; Frank and Badre, 2012) requires learning both the proper input gating strategies (which PFC stripes to gate information into, depending on the orientation, and which to continue to maintain during intervening items), and output gating strategies (which PFC stripes to read out from in response to a particular orientation probe). Such learning is acquired via reinforcement learning mediated by dopaminergic signals projecting to the striatum

(represented by the bottom half of the model). At the time of recall, the network receives a dopamine (DA) burst or dip conveying a reward prediction error signal (RPE, by comparing the reward it receives with the reward it expected to receive using a Rescorla Wagner delta rule algorithm). These DA signals are used to modulate synaptic plasticity in the Go and NoGo units, reinforcing corticostriatal Go signals that drive adaptive input and output gating operations (following positive RPEs), and reinforcing NoGo units that suppress ineffective gating strategies (following negative RPEs). (See Supplementary Materials for information on parameter searching for other parameters of the model.) The model's previous gating decisions are flagged using an eligibility trace via synaptic tagging. Thus, Rewards modulate Go and NoGo synaptic weights not only based on their current activity levels but also based on these synaptic tags reflecting recent activity.

In addition to the gating strategies, which are learned via dopaminergic reinforcement learning as per above, the network also has to learn to produce the correct color in the output layer. On each Store and Ignore trial, regardless of whether stimuli are gated into PFC, the network has to report the color presented in the input. As such the connections from the input to output layers are plastic, and this mapping is learned via supervised learning (i.e., target colors are presented and the network learns from errors using the XCAL learning rule, see Supplementary Materials). On recall trials, the network also has to learn to map the neurons that are output gated from PFC to reproduce the associated color in the output layer. As such, connections from PFC output stripes to the output layer are also plastic and learns according to the same rule. (Note that this is only useful if the corresponding stimuli have been gated in and out of PFC). All other weights are fixed (i.e., from striatum to GPiThal, and from PFC superficial to deep layers.

All networks are trained for 500 epochs and 100 trials per epoch. This was more than sufficient for learning to converge. We found that despite some fluctuations in training curves late in training, the gating strategy used by the model was largely unchanging at this point.

## Chunking / nearest neighbor implementation

The base model incorporates purely feedforward visual input that can be stored in prefrontal cortex. But it is widely appreciated that there are also top-down projections from prefrontal cortex to posterior sites (e.g., in parietal cortex). We posited that these top-down projections would allow networks to bias the sensory representation toward those stored in PFC, facilitating chunking. The degree of such bias should depend on the strength of those projections, and indeed experimental evidence shows multiple abstractions of an item across parietal cortex and auxiliary sensory regions Ito and Murray (2023).

We considered a minimal set up in which the network has access to two such representations, allowing it to represent both the raw sensory input (through connections directly to PFC) but also via a "chunking layer" representing posterior cortical association areas that receive excitatory input from both sensory regions and top-down information from deep PFC maintenance layers (Figure 1.4a). This convergent excitatory connectivity ensures that if any of the currently maintained PFC representations is close enough to the current sensory input, the overlapping neural activations will be enhanced, thereby biasing the bump attractor in the chunk layer to be attracted toward the nearest PFC representation(s). Lateral inhibition ensures that only the most excited units will remain active. As such, the chunk layer will either represent the original incoming stimulus (if no PFC representation is sufficiently close to it) or a combined representation between the incoming and existing stimuli (Figure 1.4b).

More specifically, the excitatory conductance to the chunk layer comes from both the input layer and the PFC and can be summarized in the following equation:

$$g_e(t) = \frac{1}{n} \sum_i x_i w_i \tag{1.5}$$

where $g_e(t)$ is the excitatory conductance (input) to a layer, $x_i$ is the activity of a particular

sending neuron indexed by the subscript i, $w_i$ is the synaptic weight strength that connects sending neuron i to the receiving neuron, and n is the total number of channels of that type (in this case, excitatory) across all synaptic inputs to the unit. Note that the relative strength of projections from input to PFC is scaled, with larger influences of input than PFC (to reflect e.g., number of synapses or proximity to the soma, using a relative weight scale ($w_i$) (see Supplementary Materials and *Computational Cognitive Neuroscience*, Chapter 2 (O'Reilly et al., 2024) for detailed information). This allows the chunking layer to preferentially reflect the input, subject to an attraction toward PFC representations that overlap with it (the nearest neighbor; Figure 1.4).

Lateral inhibition regulates activity in the chunking layer and, together with the strength of the top-down PFC projections, determines how close representations must be to bias the input toward the nearest PFC representations. (If the peak of the PFC bump attractor overlaps only with the tail of the sensory input bump, its influence will be largely suppressed due to inhibition). Lateral inhibition is implemented using feedforward (FF) and feedback (FB) inhibition (FFFB) which both alter the inhibitory conductance $g_i(t)$:

$$g_i(t) = \mathrm{Gi}\left[\mathrm{ff}(t) + \mathrm{fb}(t)\right] \tag{1.6}$$

,

where feedforward inhibition ($ff(t)$) is determined by the summed net input to the layer and feedback inhibition ($fb(t)$) is determined by the firing rates of the neurons in that layer, and the $Gi$ gain parameter determines the overall sparsity of the layer (i.e., the relative influence of inhibition compared to excitation $Ge$); see O'Reilly et al. (2024).

Critically, a given PFC stripe receives projections from either the sensory input or the chunking layer. (The more general idea is that PFC stripes may have access to posterior cortical layers having varying levels of top-down projections and therefore chunking

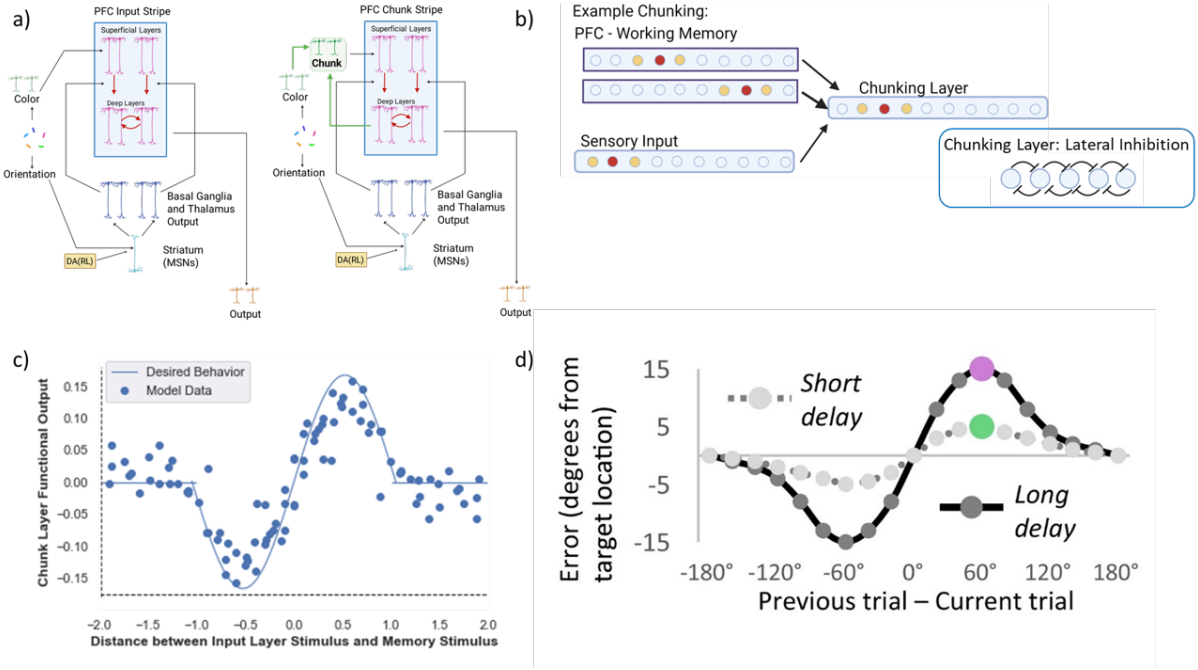



Figure 1.4: **PBWM Model and Chunking Layer Details.** a) Network diagram in the minimal case of two stripes: the first PFC stripe receives projections from the input layer ("PFC Input Stripe"); the second PFC stripe receives projections from the chunk layer ("PFC Chunk Stripe"). The network can be scaled to include more stripes of either type. We will refer to this model as the "chunk model". The control "no chunk" model has two stripes that are both of the type "PFC Input Stripe" (but it can use them to store separate inputs). b) Chunking schematic. A posterior ring attractor layer receives both bottom up sensory input and top-down input from the two PFC stripes (maintaining separate stimuli/representations). Overlap between the sensory input and the first PFC representation leads to convergent excitatory input in the chunking layer, resulting in a merged attractor. The impact of the more distant PFC representation is suppressed due to lateral inhibition. c) Chunking profile based on similarity. The x-axis shows the difference (in arbitrary units - comparable to radians) between the incoming stimulus and the nearest stimulus in PFC. The y-axis shows the deviation in the decoded chunk layer representation from the input stimulus. If the sensory input is close to a PFC representation, the chunk layer is attracted toward it. If the difference between the input stimulus and the nearest PFC representation is too large, the chunk layer largely mirrors the input (due to stronger input than PFC projections and lateral inhibition). This chunking profile closely matches that seen human memory representations, whereby memory reports are biased toward recent stimuli (top right inset, adapted from (Kiyonaga et al., 2017)).

profiles). As such, PBWM could learn a gating strategy to store either the original sensory input into the corresponding PFC stripe or to store the chunked representation into the other stripe. In the latter case, it would replace the existing item stored in that PFC stripe with the new chunked representation, incorporating the novel input stimulus. These gating strategies are not hard-wired but are learned. For instance, the network could learn to use one stripe to store colors linked to particular orientations and use the other stripe for the rest of the orientations, allowing it to appropriately manage where to store and read out information when given the probe. In this case it would have precise memory for those representations that are stored and accessed, but it would have to guess if the probed item was not stored. At the other extreme, the model could learn to preferentially gate representations into and out of the chunk stripe but with less specificity. We will see later how the model gating policy depends on task demands (specifically set size) and evolves with learning.

For each experiment, at least 80 separate random seeds were run for each network, and results are averaged across them. (For select analyses requiring more observations, 160 separate random seeds were used). To test how set size affects learning and memory performance, the models were trained and tested with set sizes 2, 3, or 4. The set size determines the maximum number of stimuli that can be presented before recall trials. For example, set size 4 means that networks have to maintain up to 4 items before receiving a recall probe, and it may have to recall any of the preceding items.

# Results

We focus our simulations on variants of the color wheel task (see Methods). Briefly, networks were presented with continuous sensory inputs represented as coarse-coded Gaussian bumps of neural activity in conjunction with their associated discrete orientation. The number of stimuli to store ("set size") before a recall trial varied between 2 and 4 items.

During a recall "probe" trial, a single orientation was presented to the network, with no color (as in the empirical versions of this task). The model had to reproduce the associated color in the output layer, in the form of a continuous population-coded response. The error is then simply the difference between the decoded color from this population and the ground truth value that was presented during the earlier store trial for that orientation. "Correct" responses show as data points when errors are close to 0 degrees. If the model outputs an incorrect color, but that color corresponds to a different orientation, this is referred to as a binding or swap error (Bays et al., 2009). Finally, if the response is incorrect and nowhere near any of the colors for the stored orientations, it would be referred to as a guess. A guess could land anywhere along the axis of -180 to 180 degrees and as such manifests as a uniform distribution across trials. If the model produced a non-response (nothing was gated out, e.g., if a stripe was empty), we randomly sampled from the uniform distribution ((Almeida et al., 2015) followed a similar process), mimicking random guessing. These errors (correct responses, guesses, binding errors) manifest themselves in the error distribution.

For proof of concept, we began with a minimal set-up in which all models were allocated 2 PFC maintenance stripes (we relax this assumption later to compare to models with larger allocated capacity). For all simulations, we compare performance (error distributions, gating strategies, influence of dopamine manipulations) between networks with chunking ability (the "chunk model") against those with equivalent (or larger) number of stripes. We refer to the "allocated capacity" as the number of stripes given to the no-chunk model, because this is a hard limit on the maximum number of representations that can be stored and accessed. We refer to the "effective capacity" as the potentially larger number of items that can be accessed due to an efficient gating policy. Effective capacity can be improved if the network learns to consistently store (input gate) colors of distinct orientations in distinct stripes, and to appropriately read out from (output gate) the corresponding stripe at recall trial. It can also potentially be

improved via chunking by increasing the number of representations that can be stored and accessed.[2]. It is thus important to note that improving effective capacity requires an effective learning process to develop adaptive gating strategies, as the networks are not hard-coded to use any stripe for storing or accessing any representation. We will show how such learning depends on a healthy dynamic range of dopaminergic RL signals in the basal ganglia.

## Error distributions across set sizes mirror those in human VWM and show chunking advantages

Figure 1.5 shows a histogram of network errors (in degrees) during recall trials. The comparisons are made between chunk and no-chunk models as well as set sizes 2, 3, and 4; in these simulations we begin with a minimal setup in which both networks are endowed with two stripes. When set size is equal to the number of stripes (2), errors are small and centered around 0, with some variance due to precision. The overall performance is similar between models, but note that the chunk model shows somewhat more imprecise responses as indicated by some more small error trials. The ability to chunk results in a small cost which manifests as a decrease in precision when chunking occurred but did not need to be used.[3].

As set size increases beyond the allocated capacity, both models resort to guessing (random errors) on a subset of trials. This pattern is observable by the error histogram containing a mixture of errors centered around zero and a uniform distribution (see Figure 1), as commonly observed in humans (Zhang and Luck, 2008). Notably, the chunking

[2]Note however, that effective capacity is not always larger than allocated capacity: without learning an effective input and output gating policy, a network's effective capacity will be less than its allocated capacity, for example if it overwrites information in the same stripe, if it accesses the incorrect stripe during recall, or if it doesn't gate a stimulus at all

[3]Note that the chunk model is endowed with two stripes, and thus the only way for it to recall both items is to use both the input stripe and the chunk stripe. As a result, at least one item could be stored precisely in the input stripe, but if the other item is close enough to it, the PFC will store the less precise chunked representation in the chunk stripe, and memory reports will be biased. The network can nevertheless store two precise items if they are far enough apart such that the chunk layer is not biased by the other PFC representation (see Figure 1.4)

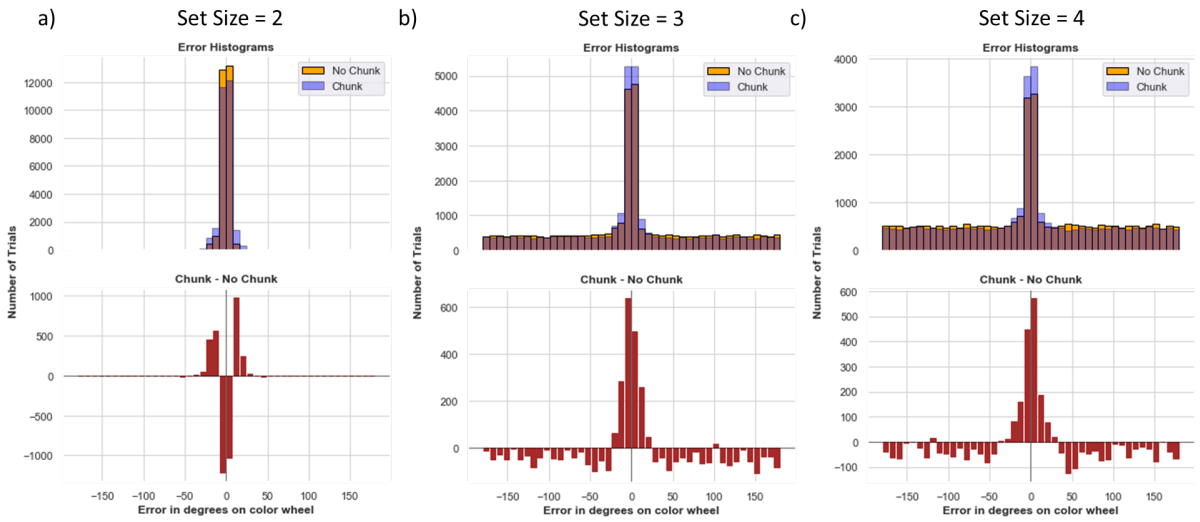
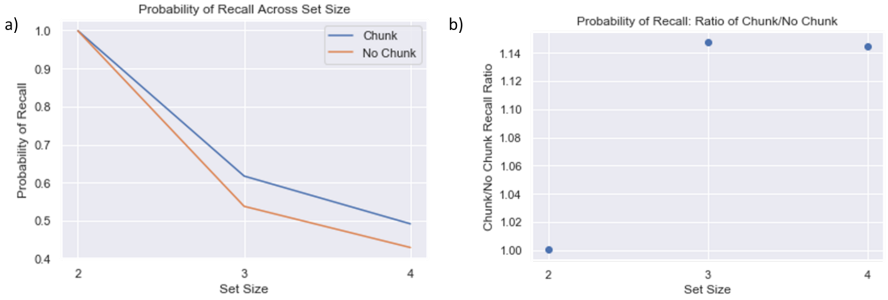



Figure 1.5: **Model Recall Error Histograms.** The binned error in degrees is plotted on the x-axis, and the number of trials for that error bin on the y-axis. The blue and orange histograms show errors from all recall trials across all 80 random weight initializations from the chunk and no chunk models, each allocated with two stripes. The red histogram plots a bin-by-bin difference in errors between the models. a) For set size 2, there is very little difference between the models. The chunk model exhibits slightly higher rates of low errors neighboring zero (up to 30 degrees), due to small losses in precision resulting from some chunking (see text). b) Set size 3 is beyond the number of stripes allotted to the network. The chunk model has a larger density at zero and small errors, and less guessing (reduced density in the uniform distribution, see red lines). c) At set size 4, the chunking advantage is manifest by low errors and the improvement in less guessing becomes more pronounced (note y-axis scale - the reduction in guessing is actually reduced for set size 4 compared to 3).

[**Model Recall Error Histograms Supplement**] **P(Recall) Across Set Size** a) Average recall probability across set sizes decreases with set size, but less so for chunk models. Note that chance performance is approximately 19%. b) Chunk models have a higher ratio of recall probability relative to no chunk model when set size exceeds allocated capacity. This analysis includes trials where the variance across colors is low (standard deviation was <35 degrees). The same chunk advantages would across all trials (including with high variance; not shown) but we focus on low variance trials wherein models can perform reasonably well even if accidentally mistaking one item for another (swap errors). Here we confirm that the chunk model improvement occurs over and above such effects.



model guesses less than the no chunk model and has a higher chance of precisely recalling the items (Fig 1.5). The difference between the chunk and no chunk model widens as the item limit continues to grow beyond the number of stripes. Comparing chunking and non-chunking models illustrates how the benefit of chunking is task-dependent. We next explored how chunking may improve the effective capacity of the network beyond the allocated number of stripes, and more specifically in which trials the advantage manifests, motivated by similar analysis in humans (Nassar et al., 2018). We subsequently explore how these strategies are acquired via reinforcement learning.

## Chunking Frees Up Space for Other Items

A key normative motivation for chunking is that doing so can save space to store other items, thereby improving effective capacity (Nassar et al., 2018). In the model, when two items are chunked into one stripe, the second stripe is free to hold another item. One key prediction is that chunking should not only manifest in terms of loss of precision when probed with any of the chunked items, but improved memory for the other items (Figure 1.2). Consider the situation in Figure 1.6 left, for a set size of 3. In both Set A and Set B¸ the red item is the probed item (the one that the model is asked to recall). Set A contains other items in the set that are different shades of green (low variance) and thus "chunkable". If the network chunks them together, they will occupy one stripe and the second stripe will be free for the red item, which is then more likely to be recalled at high precision (as it is not part of the chunk). In Set B, the other items are pink and green and are thus not chunkable (high variance). The network may store each of these into a separate stripe, forcing it to randomly guess when probed with the red stimulus. (Alternatively, the red item could be chunked with the pink item, in which case it will be recalled but with less precision than if it stored only the original red item). To formalize and test this intuition, we quantified the "out of cluster variance" (OCV) as the variance in degrees between the "other" items in the set (i.e., the items that are not probed; see

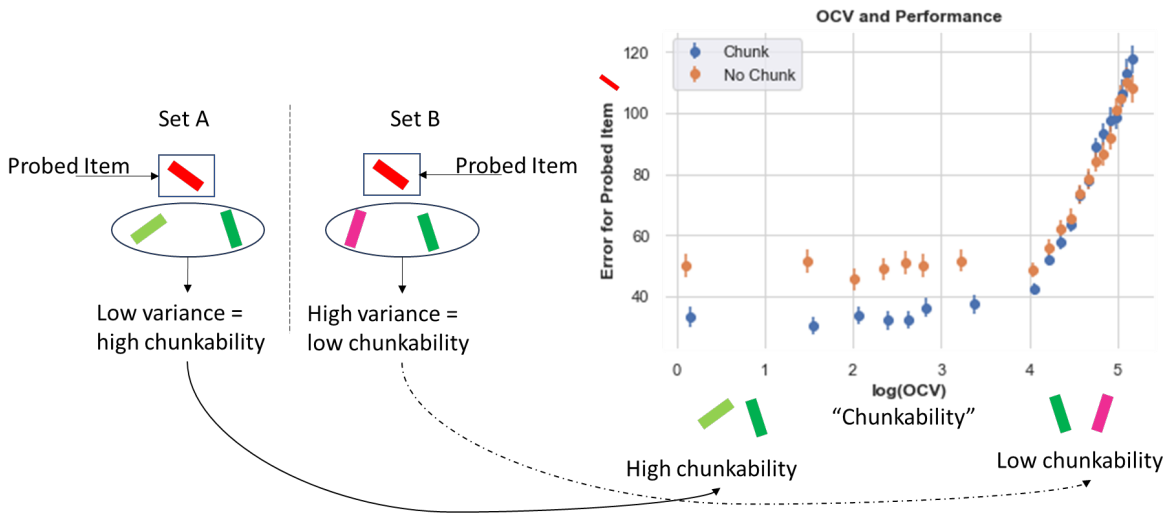



Figure 1.6: **Chunking improves recall for non-chunked items.** Left. Example array. Here we compare 2 sets, both containing a red item that will be later probed. In Set A, the other items (out of the probed cluster) are two shades of green and thus low variance (are similar to each other) and are therefore more likely to be chunked. In set B, the out of cluster variance (OCV) for the green and pink items is higher and these items are not likely to be chunked. Right. Chunking networks show consistent Recall advantages (lower errors) when OCV is low and hence the other items are chunkable. This difference disappears as OCV increases and overall errors rise. Errors plotted over all trials averaged over 80 networks in each case.

Supplementary Materials for details on OCV calculation). When this variance is low, those items are more chunkable. We then assessed whether accuracy on recalling the probed item (not part of this chunk) is improved as a proxy for chunking behavior.

Indeed, the data supports this prediction (Figure 1.6). When other items in a set are chunkable (low OCV), the chunking network exhibits significantly smaller errors than the control network on the probed item. After some OCV threshold, the chunking network no longer exhibits an advantage, and both networks show increasing errors. (The increasing errors likely result from "swap errors" (Bays et al., 2009), i.e., when the network reports one of the other stimuli in its memory instead of the probed item - this results in larger errors as the entire set is more uniformly spaced and thus not chunkable.) In sum, this analysis confirms that chunking does not merely result in reporting imprecise responses for nearby items due to perceptual similarity, but that the network leverages such similarity to its advantage so that it can save space for storing and recalling other items, as also

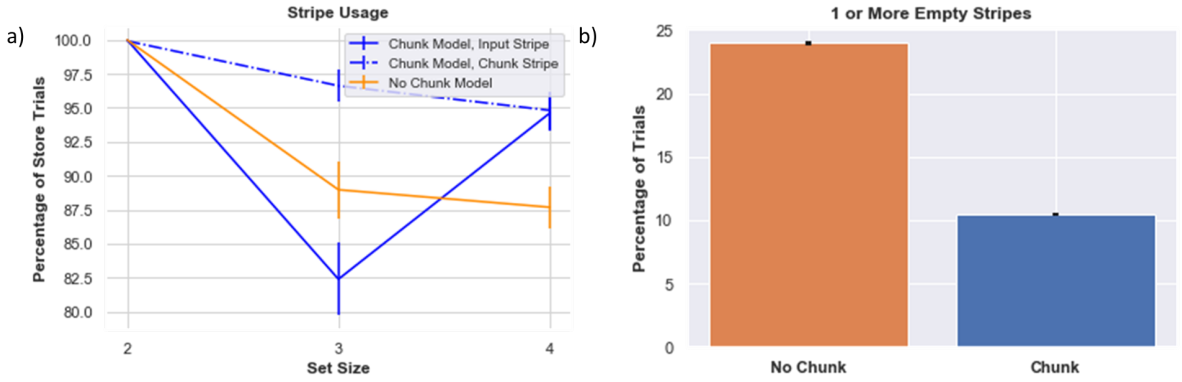



Figure 1.7: **Stripe Usage** a) Stripe usage for the 1) chunk model, chunk -linked stripe 2) chunk model, input-linked stripe 3) no chunk model (average across both stripes), b) Proportion of trials when at least one stripe was empty. This analysis was done over 160 different model initializations.

seen in humans (Nassar et al., 2018).

## Chunking leads to better resource management

In addition to overall better performance, we hypothesized that chunk networks can manage memory resources more efficiently than the no-chunk control models. We next compare how the models use the allocated resources, focusing on store trials in which the maximum number of stimuli were presented (e.g., 4 stimuli for set size 4). We begin by analyzing the performance of networks after learning; below we explore how the chunk network learns to use the different gating strategies over the course of learning for different set sizes.

When the set size is 2, after learning, chunk and no-chunk models are equally able to utilize both stripes on 100% of the trials (Figure 1.7). The networks can properly gate both colors into distinct memory slots without overwriting or reusing the same stripe (in which case Fig 1.7 would have shown reduced use of one or the other stripe).

As the set size increases beyond allocated capacity, the gating management problem becomes much harder to solve via RL, and overall performance declines, particularly in the no-chunk (control) model. Indeed, one might expect that as the number of items

are increased, the model should always "max out" the number of stripes used, but in fact the opposite is the case. When the network attempts to gate information into both stripes, the no-chunk model will often receive negative reward prediction errors during recall trials when it will inevitably be forced to guess for a subset of the stimuli that is not in its allocated memory. As a result, input gating strategies will be punished, even if they were successfully employed and would have been useful had the other stimuli been probed. In turn, due to punishing gating policies that are generally useful, the stripe usage actually decreases, and the stripes are sometimes empty, akin to "giving up". Conversely, if the network happened to be positively reinforced for gating a particular stripe, it might promiscuously gate that same stripe for other stimuli. This leads to overwriting information even though the other stripe is empty, and forcing the network to guess (or emit a non-response) when probed with a stimulus that was overwritten. In sum, the model exhibits a non-monotonic use of its resource, as its effective capacity actually declines relative to its allocated capacity. This result is reminiscent of experimental data in fMRI and EEG showing that PFC activity increases with increasing set size but then plummets when set size exceeds the participants capacity (e.g. (Zhang et al., 2016)), perhaps indicating a "giving up" strategy. We will explore the dopaminergic RL basis of such giving up in a section below.

In contrast, the chunk model is more effective at managing its resources when set size exceeds allocated capacity (Figure 1.7). Recall that the chunk model has access to one stripe with input from the chunked representation ("chunk stripe") and one stripe with raw sensory input ("input stripe"). As set size increases, the network has more opportunities for chunking, and accordingly, relatively more instances of reinforcing the gating operation linked to the chunk stripe. Interestingly, as set size just exceeds allocated capacity (here, for set size 3), the network decreases its use of the input stripe. This pattern arises because the network learns an advantage of chunking for set size 3 and thus sometimes does so more than it needs to (freeing up the input stripe), but also because of

the cost of relying too much on the input stripe (as per the no-chunk model). Finally, as set size increases further (set size 4), the chunk network learns the benefit of storing some items in the held out input stripe, increasing effective capacity, while still effectively using the chunk stripe.

In sum, this analysis suggests that resource management and utilization is more consequential than the absolute number of stripes available. In previous network models of visual WM (Wei et al., 2012; Nassar et al., 2018; Edin et al., 2009; Almeida et al., 2015), responses were considered "correct" if the probed item was stored somewhere within the network; i.e., networks were not required to select among these stored representations in response to a probe, and they also did not have to decide whether or not to store an item in the first place. In contrast, the PBWM model focuses on the control of gating strategies into and out of WM, but requires RL to do so. Errors can result from reading out the wrong stripe (a swap error) or from an empty stripe (leading to guessing or non-responding). Chunking is an information compression mechanism that allows multiple stimuli to be mapped onto the same stripe. The chunk stripe has the advantage of being used repeatedly, giving the network has more opportunities to learn how and when to use that stripe.

## Chunking advantages remain even when comparing to networks with higher allocated capacity

One might think that chunking advantages are limited to situations in which the allocated capacity is less than the set size. But when considering the challenges imposed in networks for which storage and access of items is not hard-wired, but must be learned, this is not a foregone conclusion. Indeed, above we found that the number of stimuli stored by no-chunk networks was even lower than their allocated capacity, due to RL challenges. We reasoned that such challenges could persist even when allocated capacity is increased to include or exceed the set size, due to credit assignment challenges in networks

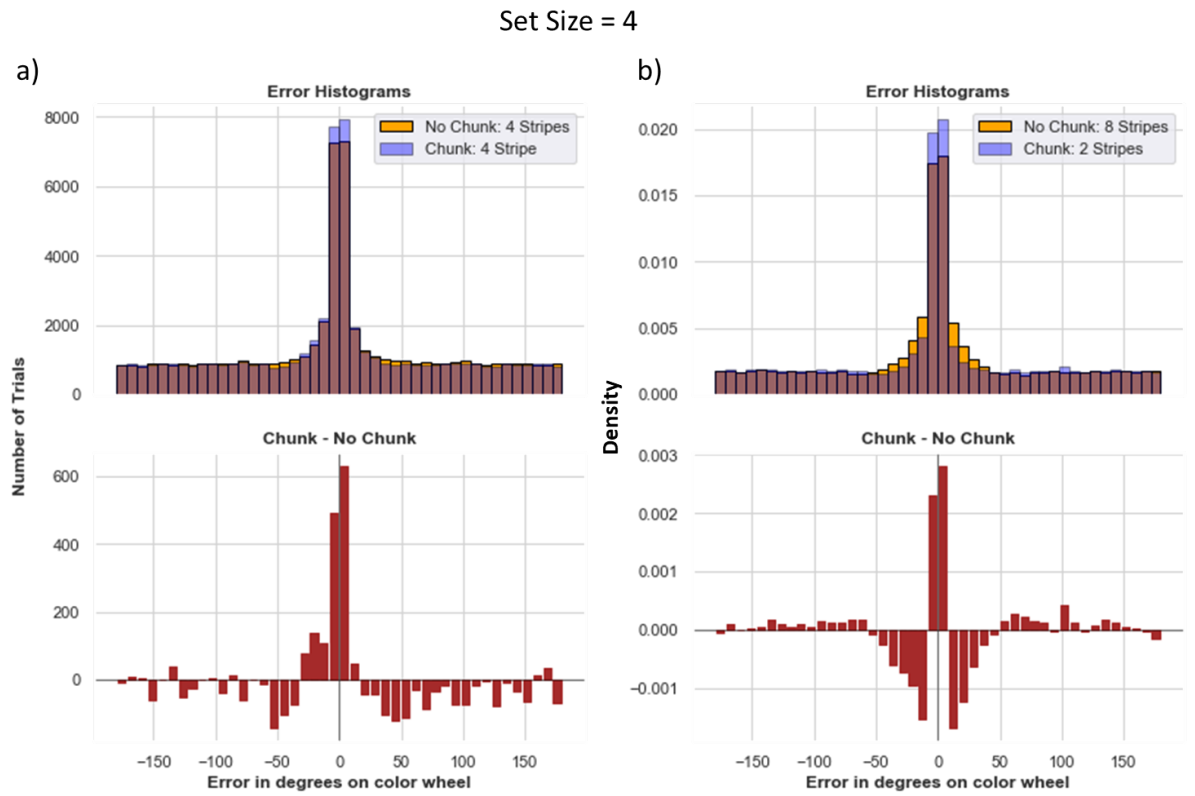



Figure 1.8: **Increasing Allocated Capacity $\neq$ Better Performance: The importance of Resource Management** a) Chunk Model with 4 stripes vs. No chunk Model with 4 stripes in a task with set size 4. Even though the no-chunk networks has sufficient number of stripes to store each item with high precision, the corresponding chunk network still exhibits advantages, due to difficulties in credit assignment associated with managing four independent stripes. b) A more extreme comparison between the Chunk Model with 2 stripes vs. No chunk Model with 8 stripes. The chunk model guesses slightly more, but has more precise responses. The 8 stripe model has more density at small nonzero errors (see text for explanation). For both a and b the averages were computed over 160 models. For b, we display density rather than counts, because trials where either models gave no response were removed to better understand the small nonzero errors in the 8 stripe model (nonresponses add noise).

that are required to learn gating strategies into and out of multiple independent PFC stripes.

We begin with an analysis of networks performing the most difficult task (set size 4) but now allocated 4 stripes (for both chunk and non-chunk networks; in this case the chunk network has just one chunk stripe and 3 input stripes). The naïve hypothesis states that no-chunk and chunk models should not differ in performance, or that chunk models might even show a deficit due to loss of precision when using the chunk stripe. Instead, based on earlier results and the above, we reasoned that chunk models might continue to show an advantage here, because frequent reinforcement of gating into and out of the chunk stripe would reduce the credit assignment burden during learning that arises from learning to manage access of 4 different items into and out of WM.

Indeed, results supported these conclusions. Error distributions in chunk networks have more density at zero and low errors. The chunking model also guesses less (Figure 1.8).

This result largely persisted even in the more extreme case when allocating the control (no-chunk) model 8 stripes (twice as many as needed) and reverting the chunk model to using only 2 stripes. While guessing is slightly reduced when having more stripes (due to more opportunities to store stimuli), the 8 stripe model does not increase the number of times it precisely recalls the correct item relative to the chunk model with only 2 stripes (Figure 1.8b). Upon further inspection, one can see that the 8 stripe model produced a larger density of moderate errors around the 0-30 degree range. This result is curious because this model has no ability to chunk. To disentangle the source of these errors and to lend insight into the difficulty of WM gating strategy with high load and/or allocated capacity, we first removed trials in which the network did not give any output (these non-responses comprised roughly 12% and 24% of trials from the 8-stripe model and the 2 stripe chunk model, respectively). Within the remaining trials, the 2-stripe chunk model has a higher peak of precise responses (around 0 error) but still slightly more guessing than

the 8 stripe no-chunk model, which continues to show a higher density at moderate errors (0-30 degrees). The reason for these moderate errors is that with 8 stripes, the network has a more difficult job. It needs to reinforce output gating strategies that properly read out from only the most appropriate stripe. Due to the curse of dimensionality (i.e., when the network outputs a response that is close enough to the target, it will get reinforced for any gating operations that preceded it, leading to spread of credit assignment to other stripes). Indeed, we found that the over-extended 8 stripe network frequently reads out from (output gates) multiple stripes in parallel (an average of 2.53 stripes during recall), and thus even when the response does reflect the correct item it is also "contaminated" by reading out one of the other stripes, such that the averaged response results in a larger number of moderate errors. In contrast, the two stripe networks (across both no chunk and chunk) output gated an average of 1.08 stripes for each trial, appropriately reading out from a single PFC representation.

In sum, simply allocating a network with larger numbers of stripes does not yield the naïve advantages one might expect, at least when gating strategies need to be learned rather than hard-wired. In this case, the networks do use all the stripes available, but don't use them effectively. For example, qualitative observations revealed that a given network might gate one single stimulus into multiple stripes, and then proceed to overwrite many or all the same stripes with a new incoming stimulus – a strategy that is sometimes effective if it happens to get probed for the one of the items still in memory during recall. The large number of input and output gating operations to consider in tandem needed for adaptive behavior leads to a vexing credit assignment problem, as it becomes a challenge to know which of several gating operations or several PFC representations are responsible for task success / error, and networks fall into an unfortunate local minimum. This credit assignment problem is mitigated by chunking, allowing the network to reinforce the same input and output gating policy across multiple instances.

## Frontostriatal Chunking Gating Policy is Optimized via RL as a Function of Task Demands

The above results show that chunking networks confer advantages as set size grows, even compared to no-chunk networks that have a larger number of allocated stripes. Moreover, these advantages come with little cost when set sizes are lower (e.g., 2). To explore how the network can adaptively learn to chunk as a function of task demands, we quantified the evolution over learning of each network's "gating policy". Prior work has shown that PBWM develops a gating policy that predicts rapid improvement in task success when such policies mimic the task structure (e.g., for hierarchical tasks (Frank and Badre, 2012); see also (Traylor et al., 2024) who showed that modern transformer neural networks mimic a PBWM gating policy when challenged with WM tasks). Here we assessed whether networks could adaptively learn a gating policy that prioritizes gating into chunk vs input stripes depending on task demands.

In PBWM and related networks, the gating policy is dictated by learned synaptic weights into striatal GABAergic medium spiny neurons (MSNs). These MSNs are classically divided into D1 "Go" MSNs and D2 "NoGo" MSNs, with opponency between these populations determining which actions are selected (i.e., those with the largest difference in Go vs NoGo activities; (Frank, 2005; Jaskir and Frank, 2023)). In the case of working memory, a network will be more likely to gate information into a particular stripe if the synaptic weights are larger for the Go in comparison to the NoGo neurons. The relative weights control the disinhibition of that particular stripe. When the network performs well and it gets a reward prediction error, dopaminergic signals modify plasticity into the corresponding D1 MSNs, reinforcing the gating policy that drove the cortical update. Conversely, errors associated with negative prediction errors lead to punishment of that gating policy by increasing synaptic weights into the D2 MSNs (Frank, 2005; O'Reilly and Frank, 2006). Below we confirm a key role for these dopaminergic signals in modulating adaptive performance. But first here we evaluated how they served to alter specific gating

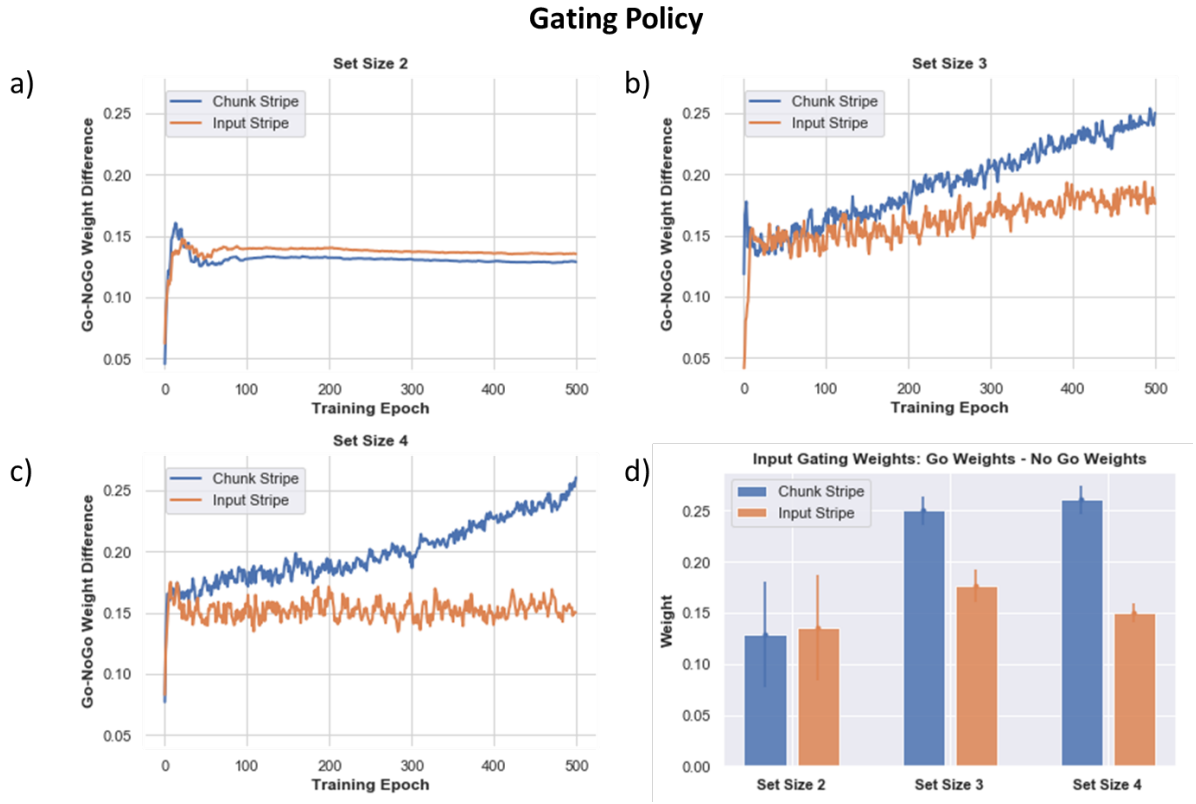



Figure 1.9: **Gating Policy (Go - NoGo Weights for Each PFC Stripe) Across Training** As the networks learn ( over 500 training epochs, averaged over 80 networks), the learned gating strategy differentiates between the input-linked (orange) or chunk-linked (blue) stripes. Positive values indicate the networks learn greater Go than NoGo weights for input gating stimuli into the corresponding stripe. a) Set size 2, the learned gating strategy shows a slight preference for the input stripe to be used (associated with increased precision), but the network also uses its chunk stripe to store the other stimulus (it is possible the chunk stripe stores a merged representation depending on the proximity of the stimuli) . b) As the set size increases to 3, the chunk stripe is increasingly preferred over training. c) This differentiation occurs earlier and more strongly for set size 4, where chunking has yet a larger advantage. d) Summary of Go - NoGo weights after training. A larger positive value shows a stronger preference for gating into that stripe. As set size increases, preference for gating into the chunk stripe increases. Relevant for training of all models: We can confirm that the network behavior has stabilized in learning even if the Go/NoGo weights continue to grow over time for the chunked layer (due to imperfect performance and reinforcement of the chunk gating strategy)

policies. We assessed PBWM gating policies in terms of the differences in Go vs NoGo synaptic weights for each stripe, and how they evolved over time when networks were trained for each set size. Specifically, we computed the summed synaptic connection strengths from the Control Input Units representing Store Orientations to the Go and NoGo input gating units in the PFC stripes corresponding to input or chunk:

$$GatingPolicy = \alpha_j = \left[\frac{\sum Go - \sum NoGo}{\sum Go + \sum NoGo}\right]_+ \tag{1.7}$$

(Here $[]_+$ indicates that only the positive part is taken; when there is less Go than NoGo, the net input to the Thalamus is 0).

If the network learns that gating information into the chunk stripe is useful, it will evolve stronger Go vs NoGo weights for that particular gating action. But if instead it is more useful to gate the veridical input stimulus, it will develop a stronger gating policy for that stripe.

Figure 1.9 shows how such gating policies evolve over time. At set size 2 - where allocated capacity (number of stripes) equals task demands, the gating policy slightly prefers to gate into the input stripe. This policy is sensible since the input stripe represents the original stimulus without any loss in precision, yielding lower errors. The network still learns a positive Go-NoGo gating policy for the chunk stripe, because it can use that to represent the other stimulus. Notably, as set size increases, the chunk stripe increasingly becomes the preferred stripe for input gating over the course of learning. This adaptive change in gating policy allows the model to optimize a tradeoff between recall quantity and precision with increasing WM load, mediating the performance advantages described above. These results also accord with those observed in humans by (Nassar et al., 2018), whereby chunking propensities evolved adaptively as a function of reward history in their experiment, and also in their meta-analysis showing that chunking benefited performance more robustly in experiments with larger set sizes.

## Dopamine Balance is Critical to Learning Optimized Gating Strategies; Implications for Patient Populations

As noted above, learning gating strategies in PBWM is dependent on the basal ganglia and dopaminergic reinforcement system. Both chunk and no-chunk networks must learn whether to gate items into WM, which stripes to gate them into so that they can be later accessed (leaving maintained information in other stripes unperturbed), and during recall, which stripe should be gated out (depending on the probed orientation). To learn this, the network uses a simple RL "critic" which computes reward expectations and deviations thereof in the form of reward prediction errors (RPEs). Positive RPEs are signaled by dopamine bursts which reinforce activity-dependent plasticity in striatal Go neurons corresponding to recent gating operations (see Supplementary Materials and O'Reilly & Frank 2006 for details). Conversely, when the model receives a reward that is worse than expected (i.e., it reports an error), a dopamine dip (a decrease in phasic dopamine) will punish previous decisions. This negative RPE will punish the gating decisions by reinforcing corresponding NoGo neurons. To assess whether a healthy balance of such dopaminergic signals are needed for adaptive gating, we manipulated the gains of these dopaminergic bursts or dips to modulate their impact on Go and NoGo Learning. These investigations are relevant for assessing the basic mechanisms of the model but may also have implications for understanding well documented working memory impairments in patients with altered striatal dopaminergic signaling, such as Parkinson's disease, ADHD and schizophrenia (Maia and Frank, 2017; Cools, 2006; Cools et al., 2007, 2008).

Figure 1.10 shows how average absolute performance across 80 networks changes with DA burst and dip gain parameters. Overall, a healthy balance of relatively symmetrical DA bursts and dips is needed for optimized performance, but this effect also interacts with set size. The best performance for set size 2 (Figure 1.10a) is along the axis where burst and dip gain are symmetrical. As set size increases, the task becomes harder, and rewards are sparser due to more errors. In this case the best performance is on the axis

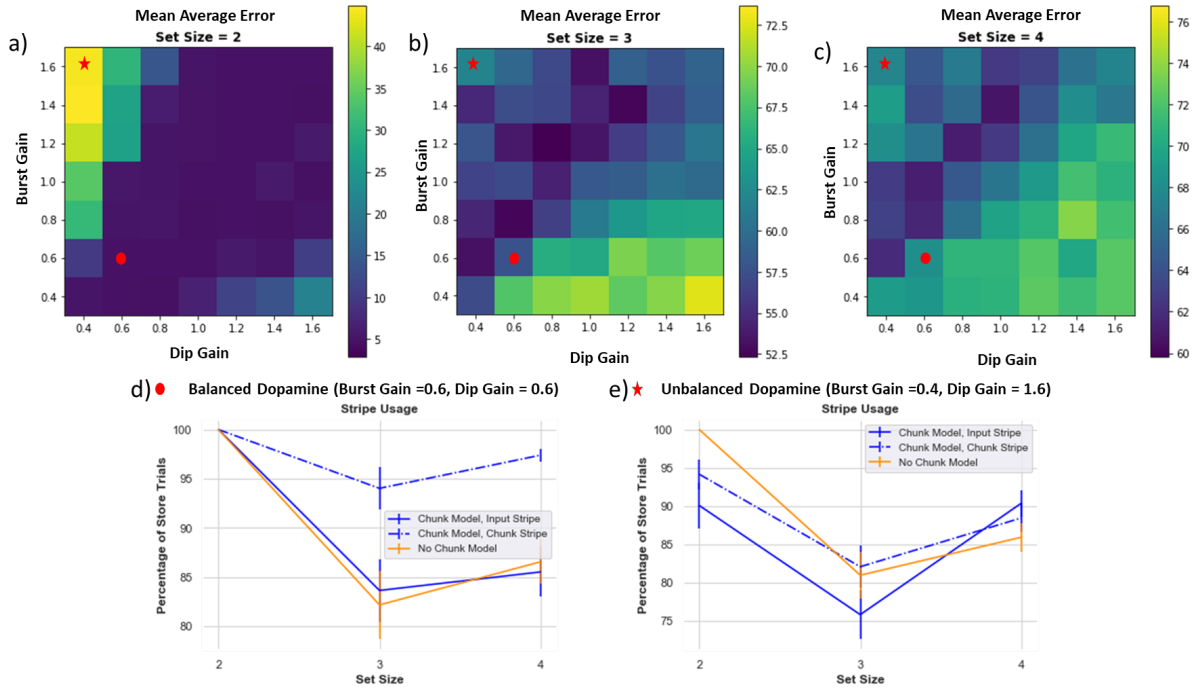



Figure 1.10: **Dynamic dopamine bursts and dips are needed for adaptive performance.** Each box is an average absolute error over 80 models. The color bar on the right indicates performance (note different scales on each plot), with darker colors (blue) representing better performance / lower absolute error. a) Set size 2: best performance is across the axis where burst and dip gain are symmetrical. b and c) Set size 3 and 4: best performance is where burst gain is slightly higher than dip gain. d) Example of balanced DA (burst gain = dip gain = 0.6), stripe usage. The chunk model manages to use the chunk stripe across all set sizes and both stripes in set size 2. The no-chunk model shows diminished storage of both stripes with increased set size due to greater propensity of DA dips. e) A regime of DA imbalance (larger DA dip than gain). The chunk model fails to robustly use both of its stripes, losing its advantage. The RL parameters interact with the ability for the chunk model to properly leverage chunking.

where burst gain is somewhat greater than the dip gain; the model learns best when it can emphasize learning from sparse rewards.

*Dopamine reinforcement signals can also lead to "giving up" and diminish effective capacity.* When set size increases, learning the proper gating strategies becomes difficult. The models may correctly gate a few items in, but they may be incorrectly overwritten or incorrectly output gated at time of recall. Importantly, incorrect responses generate negative RPEs that punish preceding gating actions, even if some of those actions were appropriate. A preponderance of negative RPEs can thus cause a network to "give up", as observed in the no-chunk models when set size exceeds allocated capacity, leading to empty stripes (Figure 1.7). This mechanism is conceptually related to rodent findings in the motor domain, whereby lower DA levels can induce aberrant NoGo learning even for adaptive actions, causing progressive impairments (Beeler et al., 2012).

*Chunking can mitigate against giving up via shared credit assignment* The chunk model can combat against using the "giving-up" strategy: when items are chunked, the chunk stripe is used more frequently and therefore has a greater chance of receiving the positive reinforcement, and thus benefits from shared credit assignment. The model still has to learn when to use the chunk vs. input stripes, but chunking serves as an aid to the reinforcement learning process. Therefore, the chunk model is also more robust compared to the no-chunk model across various parameter ranges of dopamine. However, the chunk model still fails if the DA dip value is sufficiently larger than the DA burst, for similar "giving up" reasons (Figure 1.10 c,e).

## Network recapitulates human sequential effects in working memory.

Finally, we also tested whether the model can reproduce well documented sequential effects in human working memory studies. Various findings indicate that in WM experiments, humans show higher accuracy for items most recently encountered. Moreover,

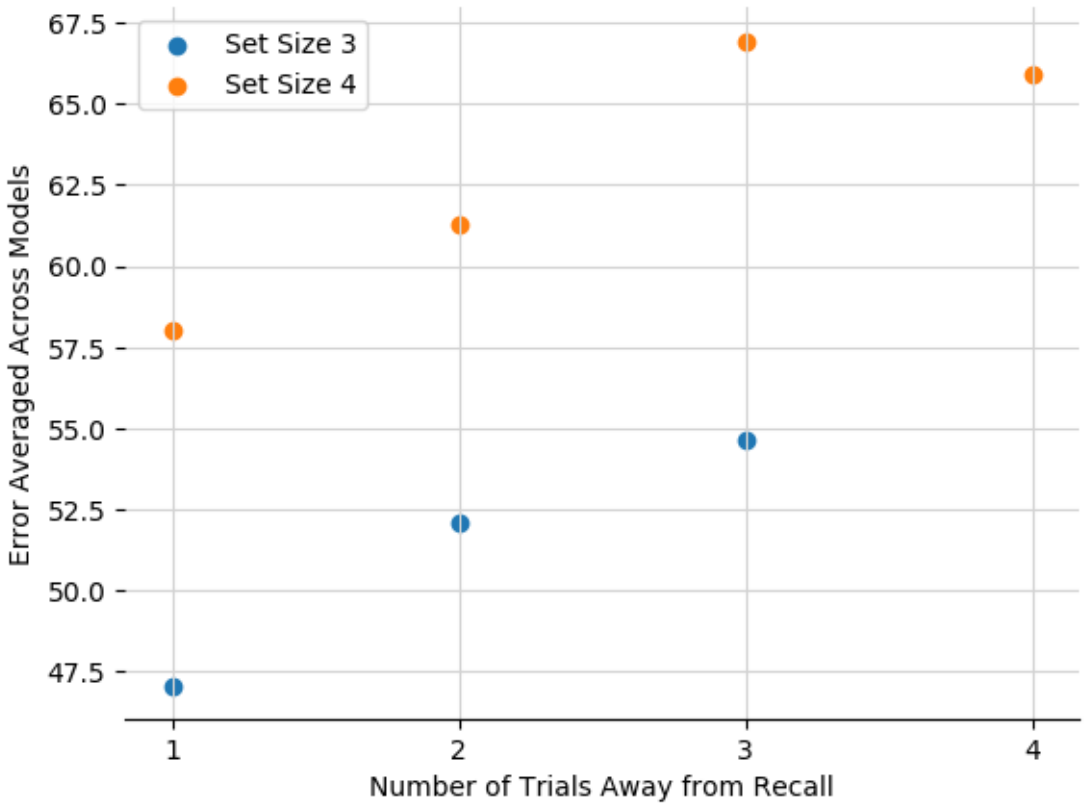



Figure 1.11: **Network Captures Recency Effects.** Average error on recall trials as a function of the distance in trials between presentation of the relevant stimulus and recall.

Oberauer et al. shows how recency effects in humans stem not just from passive decay but specifically from intervening distractors. Our network reproduces these effects, with average error monotonically increasing with number of intervening trials since the relevant stimulus was encountered, in both set sizes 3 and 4. This results from the simple principle whereby with more intervening trials, the gating model is more likely to have updated a corresponding stripe, either replacing it altogether (leading to increased probability of forgetting) or chunking with previous items, thereby increasing average error. The errors are overall smaller in set size 3 because the network is less likely to overwrite an item altogether (it can chunk and still recall one of the items perfectly). Finally, the recency effects asymptotes in set size 4 due to increased opportunities for chunking(indeed, no-chunk networks continued to show increasing errors 4 trials back; not shown).

# Discussion

Our work synthesizes and reconciles theories of visual WM across multiple levels of abstraction. At the algorithmic level, there has been extensive debate regarding the nature of WM capacity limitations, with experimental results alternately supporting slots or resources theories (Bays et al., 2009; Berg et al., 2012; Wei et al., 2012; Swan and Wyble, 2014). At the mechanistic level, several studies across species and methods suggest that circuits linking frontal cortex with basal ganglia and thalamus support WM input and output gating (McNab and Klingberg, 2008; Cools et al., 2007, 2010; Baier et al., 2010; Nyberg and Eriksson, 2016; Chatham et al., 2014; Wilhelm et al., 2023; Rikhye et al., 2018; Nakajima et al., 2019). These data accord with the PBWM and related neural network models of PFC-BG gating processes (Frank et al., 2001; O'Reilly and Frank, 2006; Hazy et al., 2007; Krueger and Dayan, 2009; Stocco et al., 2010; Badre and Frank, 2012; Calderon et al., 2022). To date, however, these lines of literature have for the most part not intersected. Here we show that when augmented with continuous population code values and a chunking layer, PBWM comprises a hybrid between slots and resources with resource-like constraints within individual stripes. Moreover, through reinforcement learning, adaptive gating policies can adjust the degree to which behavior mimics primarily slots-like or resource-like as a function task demands. As such, this model accounts for human findings supporting chunking of related items in WM, that such chunking evolves with reward feedback, and is predictive of better performance with increasing task demands across multiple datasets (Nassar et al., 2018).

On its surface, PBWM is ostensibly a slot-like model, with distinct PFC clusters ("stripes") corresponding to slots that can be independently gated, giving rise to useful computational properties such as variable binding, indirection, compositionality and hierarchical generalization (O'Reilly and Frank, 2006; Kriete et al., 2013; Collins and Frank, 2013; Calderon et al., 2022). The need for gating also accords with data suggesting that effective WM capacity is related to management of WM content, ie. one's ability to

filter out distractors so as to prioritize task-relevant information (Vogel et al., 2005; Astle et al., 2014; Feldmann-Wüstefeld and Vogel, 2019), and that such abilities rely on basal ganglia output and function (McNab and Klingberg, 2008; Baier et al., 2010). However, previous applications of PBWM and related networks have not confronted tasks requiring storage of continuous valued stimuli. In this work we augmented PBWM with a ring attractor layer that resembles that which has previously been applied to such continuous report tasks, supporting population level coding and mergers of nearby attractors (Wei et al., 2012; Edin et al., 2009; Nassar et al., 2018). However, in our network, this layer receives bottom-up input from both the sensory input layer and top-down input from all the PFC stripes, thereby allowing the network to combine sensory information with the nearest neighbor in memory. Moreover, WM chunking in our model is not obligatory, as the network can learn a gating policy that prioritizes raw sensory inputs (in which case it can represent a given item precisely) or to either replace a currently stored PFC item with the chunked version. As such, the PBWM-chunk model can learn to store more items than the allocated slots capacity by combining representations while incurring the cost of lost precision on the chunked items, giving rise to resource-like behavior. Given this learned policy, the network still may encounter trials where chunking is not possible and all stripes are occupied, leading to guessing and slots-like behavior. Depending on the learned gating policy and the task, the errors look more "slots-like" or "resource-like".

As such, our model addresses key limitations in previous neural models in this domain, in which chunking was obligatory fashion due to perceptual overlap and could not be optimized (Wei et al., 2012; Nassar et al., 2018). Instead, PBWM adapts whether or not to chunk as a function of task demands and reward history (Figure 1.9), similar to empirical data (Nassar et al., 2018). Further, PBWM can also report only the color of the probed item, unlike previous neural models which were considered accurate as long as the probed color was one of the various populations still active (Wei et al., 2012; Nassar et al., 2018).

Critically, to perform adequately, PBWM requires learning appropriate input and output gating policies which are not hard-wired, and indeed involves solving a difficult credit assignment problem (O'Reilly and Frank, 2006). At the input gating level, the network must learn whether to gate the chunked or the raw sensory stimulus (via updating of the chunk vs input stripe). Simultaneously it must also learn which stripe to output gate in response to a given probe, which requires coordinating its input and output gating strategies so that they align. The credit assignment problem, understanding which input gating decisions in combination with output gating decisions lead to reward, is difficult. To understand the difficulty, we can look at an example case where the model input gates into stripe 1. However, during read out, since it has not learned the proper binding yet, it gates out from stripe 2, leading to an error and a dopamine dip. In this case, due to an improper output gating decision, both input gating decisions and output gating decisions will be punished. Eventual successful performance requires exploration of alternate gating strategies and reinforcement of those that are effective.

How can chunking help? First it is important to note that the above problem becomes even more difficult as the number of stripes increases – even if it matches or exceeds the set size (as shown in Figure 1.8). For example, random exploration and guessing will lead to the correct response (an item being gated into a stripe AND read out from the correct stripe) 50% of the time with 2 stripes and 33% if the model has 3 stripes. The general form is: $\frac{(n-1)^{N-1}}{n^N}$ where n is the number of stripes and where N is the set size.For a quick intuition, we assume that the first item is gated into any one of the stripes. We then multiply two probabilities: 1) the probability that the second item is gated anywhere *except* where the first item was stored - which is $n - 1/n$ for 1 additional item. This probability is multiplied as many times based on the size size minus 1 since the first item is already stored (the power is $N - 1$) 2) The probability that the first item is gated out correctly, which is $1/n$. The probability of this correct guess goes down as the number of stripes increases. The difficulty also increases with set size because the network must learn

where to input and output gate for each item, and it is also possible for it to overwrite information by updating a stripe. As such, using a smaller number of stripes but allowing for chunking provides a lossy compression strategy that can mitigate this problem and render credit assignment easier, despite the loss of precision. Rather than overwriting information, the network can learn to use the chunk-linked PFC stripe if it is predictive of task success (minimal cost in the reward function for small errors), and moreover, when chunking is "good enough" the network can leverage repeated reinforcement of the same gating policy to store and read out from the chunked-link PFC stripe, thereby improving credit assignment.

As such, our simulations provide a provocative if somewhat speculative understanding of the nature of WM capacity limitations. It is unlikely that such limitations result from limits in the number of neurons or stripes available in prefrontal cortex, given that discrete estimates on capacity limitations range in the order of 3-4 items whereas the number of stripes (or equivalent clusters of PFC populations) is orders of magnitude larger (Frank et al., 2001). Our simulations show that a limiting step is properly utilizing and managing resources to optimize performance (Figure 1.7), and that it might actually be more effective to limit the number of representations used. Increasing model capacity to 4 and 8 stripes and the resulting comparisons show that the limitation in the model is not simply about number of slots but the complexity of learning. Using WM requires multiple input and output gating decisions and strategies in tandem with solving and learning a task - this would become trivial with a homunculous dictating what information to store and where to store it. In biology, the PFC has to *learn* these gating strategies: it is not hardwired. This set of experiments helps explain various other WM findings which suggest that effective WM capacity is not just about "capacity" but rather is also about the ability to filter out irrelevant information, the importance of the task (reward), and experience with the task (experts vs. novice) (McNab and Klingberg, 2008; Astle et al., 2014; Nassar et al., 2018; Feldmann-Wüstefeld and Vogel, 2019; Nakajima et al., 2019). It

also accords with our findings that when exceeding capacity, networks often "gave up" in the sense that they had more trials in which at least one stripe was empty, due to the influence of negative prediction errors punishing gating policies. As such, we showed that networks require a larger dopamine burst than dip to succeed with increasing task demands. This finding also accords with related data in rodents and our network model in the motor domain, whereby dopamine depletion can cause a progressive 'unlearning' of adaptive strategies (i.e., "giving up") via aberrant NoGo learning (Beeler et al., 2012). This learned Parkinsonism was shown to be related to plasticity in D2 medium spiny neurons (Beeler et al., 2012), and this mechanism was recently confirmed to depend on the indirect pathway (Cheung et al., 2023).

More generally, our simulations revealed an important role for RL in shaping gating policies as a function of task demands, mimicking normative analysis showing that optimal chunking criterion changes with set size (Nassar et al., 2018). In the network, dopamine is a critical component of RL by adjusting synaptic weights into striatal modules that support input and output gating. The need for a healthy balanced dynamic range of DA signals for adaptive performance provides a potential window into a mechanism that can explain deficits in patient populations with altered striatal DA signaling. Whereas much of the literature in patients with schizophrenia and ADHD focuses on limitations in WM capacity, our simulations suggest an alternative whereby altered DA signaling in these populations (Maia and Frank, 2017) could influence chunking and efficient use of resources. Our finding that adaptively learned gating policies are important for controlling when and whether to chunk may have implications for recent accounts of patient populations with repetitive negative thinking, which are proposed to arise from failures inherent to learning adaptive mental gating (Hitchcock and Frank, 2024). Future work in these patient populations could aim to study these nuances for better understanding of their cognitive deficits.

## Limitations and Future Directions

There are several limitations to this work. For simplicity, we restricted our simulations to a chunking network with just one chunk-linked PFC stripe and one or more input stripes. In this case, the determining factor for whether stimuli are merged in the chunking layer depends on how close they are in color, lateral inhibition in the chunking layer, and the relative strength of top-down PFC projections to the chunk layer. These parameters were fixed in our simulations and were not formally optimized. A more general model could include multiple chunking layers with a reservoir of effective chunking thresholds (e.g. with varying degrees of top-down influence and lateral inhibition). Depending on the task, the model could learn to chunk more liberally (larger set size - larger threshold) or more restrictively (smaller set size - smaller threshold), by adapting gating policies to rely on PFC stripes linked to these finer or coarser representations. Alternatively, it is possible that a network could learn to adapt these hyperparameters directly within the chunking layer. Further, through development the brain learns the environmental statistics and could learn those threshold parameters on a developmental time scale and could be fine-tuned on a task by task basis. Our objective was to explore how far one can get by optimizing only the gating policy via biologically plausible RL rules explored widely in basal ganglia.

Because of its wide application in the literature, we considered tasks in which stimuli can be chunked along a single scalar dimension (color or orientation, both of which have shown evidence for chunking Nassar et al. (2018)). Future work should explore to what degree these principle could generalize to more complex stimuli where chunking could occur across other more abstract dimensions, depending on the task demands (Kiyonaga et al., 2017). This model has the potential to be scaled up and here we show the core principles for how the chunking gating strategy can be learned via RL.

One key difference is how the task is presented to the model and to humans. Humans are given clear verbal instructions and are able to perform the color wheel task with little

to no practice. However, the model does not receive verbal communication and must learn the task from scratch - random weights. It has no prior experience with how to process the stimuli or how to maintain any stimuli. Humans learn this through development and establish a general gating policy. In a everyday setting, while individuals are not re-learning connections and gating policies to fit individual tasks, they are "fine-tuning" how they manipulate the information in WM and how to apply their learned policies to adapt to the current task. Experimental results show how reward or task relevance are factors that can tweak gating policies ((O'Reilly and Frank, 2006; Nassar et al., 2018)).

**Acknowledgments**

Aneri Soni was supported by NIMH training grant T32MH115895 (PI's: Frank, Badre, Moore). The project was supported by ONR MURI Award N00014-23-1-2792 and NIMH R01 MH084840-08A1. Computing hardware was supported by NIH Office of the Director grant S10OD025181.

# References


Almeida, R., Barbosa, J., and Compte, A. (2015). Neural circuit basis of visuo-spatial working memory precision: A computational and behavioral study. *Journal of Neurophysiology*, 114.

Astle, D. E., Harvey, H., Stokes, M., Mohseni, H., Nobre, A. C., and Scerif, G. (2014). Distinct neural mechanisms of individual and developmental differences in vstm capacity. *Developmental psychobiology*, 56.

Badre, D. and Frank, M. J. (2012). Mechanisms of hierarchical reinforcement learning in cortico-striatal circuits 2: Evidence from fmri. *Cerebral Cortex*, 22.

Baier, B., Karnath, H. O., Dieterich, M., Birklein, F., Heinze, C., and Müller, N. G. (2010). Keeping memory clear and stable - the contribution of human basal ganglia and prefrontal cortex to working memory. *Journal of Neuroscience*, 30.

Bays, P. M., Catalao, R. F., and Husain, M. (2009). The precision of visual working memory is set by allocation of a shared resource. *Journal of Vision*, 9.

Beeler, J. A., Frank, M. J., McDaid, J., Alexander, E., Turkson, S., Bernandez, M. S., McGehee, D. S., and Zhuang, X. (2012). A role for dopamine-mediated learning in the pathophysiology and treatment of parkinson's disease. *Cell Reports*, 2.

Berg, R. V. D., Shin, H., Chou, W. C., George, R., and Ma, W. J. (2012). Variability in encoding precision accounts for visual short-term memory limitations. *Proceedings of the National Academy of Sciences of the United States of America*, 109.

Bliss, D. P., Sun, J. J., and D'Esposito, M. (2017). Serial dependence is absent at the time of perception but increases in visual working memory. *Scientific Reports*, 7.

Brady, T. F. and Alvarez, G. A. (2015). Contextual effects in visual working memory reveal hierarchically structured memory representations. *Journal of Vision*, 15.

Brady, T. F., Konkle, T., and Alvarez, G. A. (2011). A review of visual memory capacity: Beyond individual items and toward structured representations. *Journal of Vision*, 11:4–4.

Calderon, C. B., Verguts, T., and Frank, M. J. (2022). Thunderstruck: The acdc model of flexible sequences and rhythms in recurrent neural circuits. *PLoS Computational Biology*, 18.

Chatham, C. H., Frank, M. J., and Badre, D. (2014). Corticostriatal output gating during selection from working memory. *Neuron*, 81.

Cheung, T. H., Ding, Y., Zhuang, X., and Kang, U. J. (2023). Learning critically drives parkinsonian motor deficits through imbalanced striatal pathway recruitment. *Proceedings of the National Academy of Sciences of the United States of America*, 120.

Collins, A. G. and Frank, M. J. (2013). Cognitive control over learning: Creating, clustering, and generalizing task-set structure. *Psychological Review*, 120.

Cools, R. (2006). Dopaminergic modulation of cognitive function-implications for l-dopa treatment in parkinson's disease.

Cools, R., Gibbs, S. E., Miyakawa, A., Jagust, W., and D'Esposito, M. (2008). Working memory capacity predicts dopamine synthesis capacity in the human striatum. *Journal of Neuroscience*, 28.

Cools, R., Miyakawa, A., Sheridan, M., and D'Esposito, M. (2010). Enhanced frontal function in parkinson's disease. *Brain*, 133.

Cools, R., Sheridan, M., Jacobs, E., and D'Esposito, M. (2007). Impulsive personality predicts dopamine-dependent changes in frontostriatal activity during component processes of working memory. *Journal of Neuroscience*, 27.

Cowan, N. (2008). Chapter 20 what are the differences between long-term, short-term, and working memory?

Edin, F., Klingberg, T., Johansson, P., McNab, F., Tegnér, J., and Compte, A. (2009). Mechanism for top-down control of working memory capacity. *Proceedings of the National Academy of Sciences of the United States of America*, 106.

Feldmann-Wüstefeld, T. and Vogel, E. K. (2019). Neural evidence for the contribution of active suppression during working memory filtering. *Cerebral Cortex*, 29.

Fischer, J. and Whitney, D. (2014). Serial dependence in visual perception. *Nature Neuroscience*, 17:738–743.

Frank, M. J. (2005). Dynamic dopamine modulation in the basal ganglia: A neurocomputational account of cognitive deficits in medicated and nonmedicated parkinsonism. *Journal of Cognitive Neuroscience*, 17.

Frank, M. J. and Badre, D. (2012). Mechanisms of hierarchical reinforcement learning in corticostriatal circuits 1: Computational analysis.

Frank, M. J. and Claus, E. D. (2006). Anatomy of a decision: Striato-orbitofrontal interactions in reinforcement learning, decision making, and reversal. *Psychological Review*, 113.

Frank, M. J. and Fossella, J. A. (2011). Neurogenetics and pharmacology of learning, motivation, and cognition.

Frank, M. J., Loughry, B., and O'Reilly, R. C. (2001). Interactions between frontal cortex and basal ganglia in working memory: A computational model.

Fukuda, K., Awh, E., and Vogel, E. K. (2010). Discrete capacity limits in visual working memory.

Hazy, T. E., Frank, M. J., and O'Reilly, R. C. (2007). Towards an executive without a homunculus: Computational models of the prefrontal cortex/basal ganglia system. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362.

Hazy, T. E., Frank, M. J., and O'reilly, R. C. (2021). Computational neuroscientific models of working memory.

Hitchcock, P. F. and Frank, M. J. (2024). The challenge of learning adaptive mental behavior. *Journal of Psychopathology and Clinical Science*.

Ito, T. and Murray, J. D. (2023). Multitask representations in the human cortex transform along a sensory-to-motor hierarchy. *Nature Neuroscience*.

Jaskir, A. and Frank, M. J. (2023). On the normative advantages of dopamine and striatal opponency for learning and choice. *eLife*, 12.

Kiyonaga, A., Scimeca, J. M., Bliss, D. P., and Whitney, D. (2017). Serial dependence across perception, attention, and memory.

Kriete, T., Noelle, D. C., Cohen, J. D., and O'Reilly, R. C. (2013). Indirection and symbol-like processing in the prefrontal cortex and basal ganglia. *Proceedings of the National Academy of Sciences of the United States of America*, 110.

Krueger, K. A. and Dayan, P. (2009). Flexible shaping: How learning in small steps helps. *Cognition*, 110.

Levitt, J. B., Lewis, D. A., Yoshioka, T., and Lund, J. S. (1993). Topography of pyramidal neuron intrinsic connections in macaque monkey prefrontal cortex (areas 9 and 46). *Journal of Comparative Neurology*, 338.

Luck, S. J. and Vogel, E. K. (2013). Visual working memory capacity: From psychophysics and neurobiology to individual differences.

Ma, W. J., Husain, M., and Bays, P. M. (2014). Changing concepts of working memory.

Maia, T. V. and Frank, M. J. (2017). An integrative perspective on the role of dopamine in schizophrenia.

McNab, F. and Klingberg, T. (2008). Prefrontal cortex and basal ganglia control access to working memory. *Nature Neuroscience*, 11.

Moscovitch, M. and Winocur, G. (2009). *The Frontal Cortex and Working with Memory*.

Moustafa, A. A., Sherman, S. J., and Frank, M. J. (2008). A dopaminergic basis for working memory, learning and attentional shifting in parkinsonism. *Neuropsychologia*, 46.

Nakajima, M., Schmitt, L. I., and Halassa, M. M. (2019). Prefrontal cortex regulates sensory filtering through a basal ganglia-to-thalamus pathway. *Neuron*, 103.

Nassar, M. R., Helmers, J. C., and Frank, M. J. (2018). Chunking as a rational strategy for lossy data compression in visual working memory. *Psychological Review*, 125.

Nyberg, L. and Eriksson, J. (2016). Working memory: Maintenance, updating, and the realization of intentions. *Cold Spring Harbor Perspectives in Biology*, 8.

Oberauer, K. (2013). The focus of attention in working memory—from metaphors to mechanisms. *Frontiers in Human Neuroscience*.

Oberauer, K., andSimon Farrell, S. L., Jarrold, C., and Greaves, M. (2012). Modeling working memory: An interference model of complex span. *Psychonomic Bulletin Review*.

Oberauer, K. and Lin, H.-Y. (2017). An interference model of visual working memory. *Psychological Review*.

O'Reilly, R. C. and Frank, M. J. (2006). Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Computation*, 18.

O'Reilly, R. C., Munakata, Y., Frank, M. J., Hazy, T. E., and Contributors (2024). *Computational Cognitive Neuroscience*. Online Book, 5th Edition, URL: https://compcogneuro.org.

Pucak, M. L., Levitt, J. B., Lund, J. S., and Lewis, D. A. (1996). Patterns of intrinsic and associational circuitry in monkey prefrontal cortex. *Journal of Comparative Neurology*, 376.

Pusch, R., Packheiser, J., Azizi, A. H., Sevincik, C. S., Rose, J., Cheng, S., Stüttgen, M. C., and Güntürkün, O. (2023). Working memory performance is tied to stimulus complexity. *Communications Biology*, 6.

Rikhye, R. V., Gilra, A., and Halassa, M. M. (2018). Thalamic regulation of switching between cortical representations enables cognitive flexibility. *Nature Neuroscience*, 21.

Stocco, A., Lebiere, C., and Anderson, J. R. (2010). Conditional routing of information to the cortex: A model of the basal ganglia's role in cognitive coordination. *Psychological Review*, 117.

Swan, G. and Wyble, B. (2014). The binding pool: A model of shared neural resources for distinct items in visual working memory. *Attention, Perception, and Psychophysics*, 76.

Traylor, A., Merullo, J., Frank, M. J., and Pavlick, E. (2024). Transformer mechanisms mimic frontostriatal gating operations when trained on human working memory tasks.

van den Berg, R., Awh, E., and Ma, W. J. (2014). Factorial comparison of working memory models. *Psychological Review*, 121.

Vogel, E. K., McCollough, A. W., and Machizawa, M. G. (2005). Neural measures reveal individual differences in controlling access to working memory. *Nature*, 438.

Wei, Z., Wang, X. J., and Wang, D. H. (2012). From distributed resources to limited slots in multiple-item working memory: A spiking network model with normalization. *Journal of Neuroscience*, 32.

Wilhelm, M., Sych, Y., Fomins, A., Warren, J. L. A., Lewis, C., Capdevila, L. S., Boehringer, R., Amadei, E. A., Grewe, B., O'Connor, E. C., Hall, B. J., and Helmchen,

F. (2023). Striatum-projecting prefrontal cortex neurons support working memory maintenance. *Nature Communications*, 14.

Xiao, Y., Wang, Y., and Felleman, D. J. (2003). A spatially organized representation of colour in macaque cotical area v2. *Nature*, 421.

Zhang, D., Zhao, H., Bai, W., and Tian, X. (2016). Functional connectivity among multi-channel eegs when working memory load reaches the capacity. *Brain Research*, 1631.

Zhang, W. and Luck, S. J. (2008). Discrete fixed-resolution representations in visual working memory. *Nature*, 453.

Adaptive chunking improves effective working memory capacity in a prefrontal cortex and basal ganglia circuit

Supplementary Materials

Aneri Soni and Michael J Frank

## 1.2 Neural model simulations

For *each frontal stripe* , the corresponding striatal gating layers consisted of 24 distributed units (12 Go and 12 NoGo) which learn the probability of obtaining a reward if the stimulus in question is gated into, or out of, it's respective working memory stripe. The number of units is proportional to the number of stripes and therefore in the model iterations with 4 or 8 stripes, the striatal gating layers are larger. In each module, a Gpi/Thal (globus pallidas internus/thalamus) unit implements a gating signal and is activated when relatively more striatal Go than NoGo units are active (subject to inhibitory competition from other GPi/Thal units that modulate gating of neighboring stripes (O'Reilly and Frank, 2006). Thus the GPi/Thal units summarize the contributions of multiple interacting layers that implement gating among the globus pallidus, subthalamic nucleus, and thalamus as simulated in more detailed networks of a single BG circuit (Frank and Claus, 2006), in these larger-scale networks we abstract away from these details. For input-gating circuits, GPi/Thal activation induces maintenance of activation states in the corresponding frontal maintenance layer (Frank et al., 2001; O'Reilly and Frank, 2006). For output-gating circuits, the GPi/Thal activation results in information flow from the frontal maintenance layer to the frontal output layer. This output layer projects to the decision circuit, such that only output-gated representations influence response selection.

The task in this experiment was the color visual working memory task. Each trial consisted of stimulus presentation, during which stimuli could be gated into corresponding

PFC areas, followed by another phase in which all input stimuli were removed and the network had to rely on maintained PFC representations in order to respond. The frontal stripes for each of the stimulus dimensions could independently maintain representations of these stimulus dimensions in $PFC_{MaintDeep}$, subject to gating signals from the BG. Initially, a "Go bias" encourages exploratory updating (and subsequent maintenance) due to novelty; these gating signals are then reinforced to the extent that the frontal representations come to be predictive of reward O'Reilly and Frank (2006). However, not all maintained $PFC_{MaintDeep}$ representations influence decision in the response circuitry, only those that are also represented in $PFC_{Out}$ due to output gating signals. Thus in a given trial, color of the current stimulus may be represented in $PFC_{MaintDeep}$, but the output gating will dictate the ultimate model response.

## 1.3 Neural model implementational details

The model is implemented using the Leabra framework (O'Reilly and Munakata, 2000, 2019), with the new version of the emergent neural simulation software, which contains extensive documentation and examples that can be run in Python or the Go language (https://github.com/emer/emergent). All of the computational models, and the code to perform the analysis, are available and will be published on our github account. The PBWM network used here simulates the anatomical projections and physiological properties of the BG circuitry in learning, working memory and decision making (Frank, 2005; O'Reilly and Frank, 2006). Leabra uses point neurons with excitatory, inhibitory, and leak conductances contributing to an integrated membrane potential, which is then thresholded and transformed to produce a rate code output communicated to other units. Dopamine in the BG modifies activity in Go and NoGo units in the striatum, and this modulation of activity affects both the propensity for overall gating (Go relative to NoGo activity), and activity-dependent plasticity that occurs during reward prediction errors (Frank, 2005; Wiecki et al., 2009; Beeler et al., 2012; Jaskir and Frank, 2023). Both of

these functions are detailed below.

The membrane potential $V_m$ is updated as a function of ionic conductances $g$ with reversal (driving) potentials $E$ according to the following differential equation:

$$\begin{aligned} C_m \frac{dV_m}{dt} &= g_e(t)\bar{g}_e(E_e - V_m) + \\ & \quad g_i(t)\bar{g}_i(E_i - V_m) + \\ & \quad g_l(t)\bar{g}_l(E_l - V_m) + \\ & \quad ..., \end{aligned} \tag{1.8}$$

$$\tag{1.9}$$

where $C_m$ is the membrane capacitance and determines the time constant with which the voltage can change, and subscripts $e$, $l$ and $i$ refer to excitatory, leak, and inhibitory channels respectively (and "..." refers to the possibility of adding other channels implementing neural accommodation and hysteresis). Following electrophysiological convention, the overall conductance for each channel $c$ is decomposed into a time-varying component $g_c(t)$ computed as a function of the dynamic state of the network, and a constant $\overline{g_c}$ that controls the relative influence of the different conductances. The equilibrium potential can be written in a simplified form by setting the excitatory driving potential ($E_e$) to 1 and the leak and inhibitory driving potentials ($E_l$ and $E_i$) of 0:

$$V_m^{\infty} = \frac{g_e \overline{g_e}}{g_e \overline{g_e} + g_l \overline{g_l} + g_i \overline{g_i}} \tag{1.10}$$

which shows that the neuron is computing a balance between excitation and the opposing forces of leak and inhibition. The excitatory net input/conductance $g_e(t)$ is computed as the proportion of open excitatory channels as a function of sending activations times the weight values:

$$g_e(t) = \langle x_i * wi \rangle = \frac{1}{n} \sum_i x_i w_i \tag{1.11}$$

The inhibitory conductance is computed as described in the next section, and leak is a constant.

Activation communicated to other cells ($y_j$) is a thresholded ($\Theta$) sigmoidal function of the membrane potential with gain parameter $\gamma$:

$$y = \frac{1}{\left(1 + \frac{1}{\gamma[g_e - g_e^{\Theta}]_+}\right)} \tag{1.12}$$

where $g_e^{\Theta}$ is the level of excitatory input conductance that would put the equilibrium membrane potential right at the firing threshold $\Theta$ and depends on the level of inhibition and leak.

$$g_e^{\Theta} = \frac{g_i(E_i - \Theta) + g_l(E_l - \Theta)}{\Theta - E_e}$$

### 1.3.1 Inhibition Within Layers

For within layer lateral inhibition, Leabra uses feedforward and feedback (FFFB) inhibition, allowing the $gi$ values for each neuron to be adjusted as a function of total net input to a layer (feedforward inhibition) as well as the total excitatory activity of that layer (feedback inhibition). The average net input (excitatory conductance) to a layer is just the average of the of each unit indexed by in the layer:

$$\langle g_e \rangle = \sum_n \frac{1}{n} ge_i$$

Similarly, the average activation is just the average of the activation values ($y_i$):

$$\langle y \rangle = \sum_n \frac{1}{n} y_i$$

The overall inhibitory conductance is just the sum of the two terms (ff and fb), with

an overall inhibitory gain constant factor Gi:

$$g_i(t) = \text{Gi}\,[\text{ff}(t) + \text{fb}(t)]$$

This Gi factor is typically the only parameter manipulated to determine overall layer activity level. The default value is 1.8 (but is reduced in the chunk layer, see below).

The feedforward (ff) term is:

$$\text{ff}(t) = \text{FF}\,[\langle g_e \rangle - \text{FF0}]_+$$

where FF is a constant gain factor for the feedforward component (set to 1.0 by default), and FF0 is a constant offset (set to 0.1 by default).

The feedback (fb) term is:

$$\text{fb}(t) = \text{fb}(t-1) + dt\,[\text{FB}\langle y \rangle - \text{fb}(t-1)]$$

where FB is the overall gain factor for the feedback component (0.5 default), dt is the time constant for integrating the feedback inhibition (0.7 default), and the t-1 indicates the previous value of the feedback inhibition.

### 1.3.2 Connectivity

The connectivity of the BG network is critical, and is thus summarized here (see Frank, 2006 and O'Reilly & Frank, 2006 for details and references). Unless stated otherwise, projections are fully connected (that is all units from the source region target the destination region, with a randomly initialized synaptic weight matrix). However the units in PFC, Striatum, GPiThal are all organized with columnar structure. Units in the first stripe of PFC represent one set of representations and project to a single column of Go and NoGo units in the Striatum, which in turn project to the corresponding column

in GPiThal. Each Thalamic unit is reciprocally connected with the associated column in PFC. This connectivity is similar to that described by anatomical studies, in which the same cortical region that projects to the striatum is modulated by the output through the BG circuitry and Thalamus.

Dopamine units in the SNc project to the entire Striatum, but with different projections to encode the effects of D1 receptors in Go neurons and D2 receptors in NoGo neurons. With increased dopamine, Go units are excited while NoGo units are inhibited, and vice-versa with lowered dopamine levels. The particular set of units that are impacted by dopamine is determined by those receiving excitatory input from sensory cortex and PFC. Thus dopamine modulates this activity, thereby affecting the relative balance of Go vs NoGo activity in those units activated by cortex. This impact of dopamine on Go/NoGo activity levels influences both the propensity for gating (during response selection) and learning, as described next.

### 1.3.3 Learning

Learning in the model is activity dependent and using a biologically motivated homeostatic learning rule called the eXtended Contrastive Attractor Learning (XCAL) rule. The empirical learning function (called the XCAL dWt function) approximates that observed from a highly biologically detailed computational model of the known synaptic plasticity mechanisms, by Urakubo et al. (2008); see Randall C. O'Reilly and Contributors (2012). XCAL uses a simple piecewise-linear function, described below, that emerges from it. This XCAL dWt function resembles the BCM learning function, where weight changes are a function of presynaptic activation $x$ and postsynaptic activation $y$ relative to a floating threshold (approximating effects of calcium levels), and is functionally similar to contrastive Hebbian learning. The floating threshold determines the amount of activity needed to elicit LTP vs LTD.

$$\Delta w = f_{xcal}\left(xy, \theta_p\right)$$

$$f_{xcal}(xy, \theta_p) = \begin{cases} (xy - \theta_p) & \text{if } xy > \theta_p\theta_d \\ -xy(1 - \theta_d)/\theta_d, & \text{otherwise} \end{cases}$$

where $\theta_p$ is the floating threshold and $\theta_d = 0.1$ is a constant that determines the point where the function reverses direction (i.e., back toward zero within the weight decrease regime).

In XCAL, error-driven learning is accommodated by allowing the floating threshold to vary as a function of recent activity (Randall C. O'Reilly and Contributors, 2012). Weights are increased if activity states during the outcome are greater than their recent levels (i.e. activity states while the network is generating a response), and conversely, weights decrease if the activity levels go down relative to prior states. Thus, we can think of the recent activity levels (the threshold) as reflecting expectations which are subsequently compared to actual outcomes, with the difference (or "error") driving learning. Thus XCAL is closely related to contrastive Hebbian learning, where weight changes are determined by changes in activation. As we will see below, in PBWM and BG models, the error in gating signals is driven by changes in activation resulting from dopaminergic RPEs (Frank, 2005; O'Reilly and Frank, 2006; Frank and Badre, 2012).

### Striatal Learning Function

Synaptic connection weights in striatal units were trained using a reinforcement learning version of Leabra. The learning algorithm involves two phases, and is more biologically plausible than standard error backpropagation. In the *minus phase*, the network settles into activity states based on input stimuli and its synaptic weights, ultimately "choosing" a response. In the *plus phase*, the network resettles in the same manner, with the only

difference being a change in simulated dopamine: an increase of SNc unit firing for positive reward prediction errors, and a decrease for negative prediction errors (Frank, 2005; O'Reilly and Frank, 2006). This change in dopamine modifies Go and NoGo activity levels, and because synaptic strengths are adjusted as a function of activity levels relative to the floating threshold (previous activity levels prior to the RPE), this functionality also influences what is learned.

For the large-scale BG-PFC models used here and in O'Reilly and Frank (2006) some abstractions are used. Each stripe (group of units) in the Striatum layer is divided into Go vs. NoGo in an alternating fashion. The DA input from the SNc modulates these unit activations in the update phase by providing extra excitatory current to Go and extra inhibitory current to the NoGo units in proportion to the positive magnitude of the DA signal, and vice-versa for negative DA magnitude. This reflects the opposing influences of DA on these neurons (Frank, 2005; Gerfen, 2000; Shen et al., 2008). The update phase DA signal reflects the critic system's reward prediction error (RPE) produced by gating signals (see below) – that is, if the PFC state is predictive of reward, the striatal units will be reinforced. Learning on weights into the Go/NoGo units is based on the activation delta that results from this RPE using the same XCAL dWt function defined above (which is functionally similar to contrastive Hebbian learning).

**Dopamine and prediction errors in the "critic"**

We used a simplified version of the critic in these simulations because they do not depend on the differences between different algorithms (e.g, temporal difference learning or "PVLV", the algorithm used in our other BG-PFC networks (O'Reilly and Frank, 2006; Hazy et al., 2007)). The algorithm we used for generating dopaminergic prediction errors corresponds to a basic Rescorla-Wagner delta rule, as also reported in Frank and Badre (2012). The use of a simple delta rule allowed us to confirm that simulation results do not depend on the details of the basic RL algorithm. The reward prediction error (RPE or

$\delta$) is the difference between the delivered reward (R) and the expected reward (V). The expected reward is calculated by a RewardPrediction unit which calculates the value of the gating actions based on the previously earned rewards.

$$\delta = R - V \tag{1.13}$$

An updated V in this RewardPrediction unit is calculated after the RPE is determined, with learning rate $\alpha$:

$$V_{updated} = V + \alpha(\delta) \tag{1.14}$$

**GPiThal Units**

The GPiThal units provide a simplified version of the GPi/Thalamus layers abstracted from the full circuitry implemented in more basic versions of the BG circuit (Frank and Claus, 2006, e.g.,). They receive a net input that reflects the normalized Go - NoGo activations in the corresponding Striatum stripe:

$$\alpha_j = \left[\frac{\sum Go - \sum NoGo}{\sum Go + \sum NoGo}\right]_+ \tag{1.15}$$

(where $[]_+$ indicates that only the positive part is taken; when there is more NoGo than Go, the net input is 0). This net input then drives standard Leabra point neuron activation dynamics, with FFFB inhibitory competition dynamics that cause stripes to compete for both input and output gating of PFC. This dynamic is consistent with the notion that competition/selection takes place primarily in the smaller GP/GPi areas, and not much in the much larger striatum e.g. (Jaeger et al., 1994). The resulting GPiThal activation then provides the gating update signal to the PFC: if the corresponding GPiThal unit is active (above a minimum threshold; .1), then active maintenance currents in the PFC are toggled.

**PFC Maintenance**

PFC active maintenance is supported in part by excitatory ionic conductances that are toggled by Go firing from the GPiThal layers. This is implemented with an extra excitatory ion channel in the basic $V_m$ update equation (1.9). This channel has a conductance value of .5 when active. See (Frank et al., 2001) for further discussion of this kind of maintenance mechanism, which has been proposed by several researchers (e.g.) (Lisman et al., 1998; Dilmore et al., 1999; Gorelova and Yang, 2000; Durstewitz et al., 2000). The first opportunity to toggle PFC maintenance occurs at the end of the first plus phase, and then again at the end of the second plus phase (third phase of settling). Thus, a complete update can be triggered by two Go's in a row, and it is almost always the case that if a Go fires the first time, it will fire the next, because Striatum firing is primarily driven by sensory inputs, which remain constant.

### 1.3.4 Continuous Stimuli

Previous applications of PBWM consisted of discrete stimuli. For this project, the stimuli were made continuous on a ring from 0 to 360 to mimic the color wheel. The Gaussian bump width is 0.15 and there are 20 units per layer.

### 1.3.5 Hyperparameter Search

The PBWM is a well established framework with many parameters. To maintain consistency with prior work and avoid an overly complex parameter search, default parameters were used with the following exceptions. A hyperparameter search was optimized for baseline model performance for parameters most relevant to this project: chunking layer inhibition (1.4), chunking layer gain (5), relative weight scales (connectivity strength) between the chunk layer and PFC layer (0.8), input layer to the chunk layer (0.7), and PFC layer to chunk layer (0.2; this lower value ensures that the chunk layer primarily reflects the input and is only influence by PFC if the nearest neighbor overlaps

with the input). The striatal learning rate was set to default of 0.04 for set size 2 and 4, but changed to 0.06 for set size 3, which improved performance for the OCV analysis in that setsize (but overall performance is not substantially altered)). The relative impact of striatal NoGo vs Go activity on GpiThal layer for inducing a gating signal was set to 1.25. Finally to accommodate continuous representations with large Gaussian bump widths in PFC we increased the PFC gain parameter to 5 (otherwise the PFC would not maintain units with lesser activation on the tails of the continuous distribution). The reward function was also altered due to the continuous nature of the outputs, such that rewards are determined based on decoded values of the output layer across the population code relative to the true color that should have been reported, as a continuous linear function of the error. We also varied dopamine burst and dip gain across a wide range to explore its impact as described in the main text.

## 1.4 Neural Model Output Analysis

### 1.4.1 Out of Cluster Variance (OCV) Analysis

The OCV analysis was performed using set size of 3, largely for interpretability. (This analysis becomes convoluted when increasing the set size: the combinations of possible chunking make it hard to divide the stimuli into "other stimuli" and "probed stimuli".)

The OCV analysis was performed on a subset of trials to specifically test whether the network can leverage chunking abilities to free up resources for other items outside of the chunk. In these trials the "chunkable" items were presented first and were within 20 degrees away from each other (in color space). The third item is at least 50 degrees away from one of the stimuli. The same procedure was applied to the no-chunk control. If the network chunks the first two items it should then be more likely to store and recall the third item.

The OCV of a single trial is the variance (across the color dimension) of all stimuli in

the trial except for the probed item (Nassar et al., 2018). The OCV is calculated for each trial and plotted against error on that trial. Since the stimuli and trials are randomly generated, some parts of the graph will be more densly populated. To combat this issue, the trials were binned so that each bin has an equal number of trials and the graph has 20 bins.

# References

Beeler, J. A., Frank, M. J., McDaid, J., Alexander, E., Turkson, S., Bernandez, M. S., McGehee, D. S., and Zhuang, X. (2012). A role for dopamine-mediated learning in the pathophysiology and treatment of parkinson's disease. *Cell Reports*, 2.

Dilmore, J. G., Gutkin, B. S., and Ermentrout, G. B. (1999). Effects of dopaminergic modulation of persistent sodium currents on the excitability of prefrontal cortical neurons: A computational study. *Neurocomputing*, 26-27.

Durstewitz, D., Seamans, J. K., and Sejnowski, T. J. (2000). Neurocomputational models of working memory. *Nature Neuroscience*, 3.

Frank, M. J. (2005). Dynamic dopamine modulation in the basal ganglia: A neurocomputational account of cognitive deficits in medicated and nonmedicated parkinsonism. *Journal of Cognitive Neuroscience*, 17.

Frank, M. J. and Badre, D. (2012). Mechanisms of hierarchical reinforcement learning in corticostriatal circuits 1: Computational analysis.

Frank, M. J. and Claus, E. D. (2006). Anatomy of a decision: Striato-orbitofrontal interactions in reinforcement learning, decision making, and reversal. *Psychological Review*, 113.

Frank, M. J., Loughry, B., and O'Reilly, R. C. (2001). Interactions between frontal cortex and basal ganglia in working memory: A computational model.

Gerfen, C. R. (2000). Molecular effects of dopamine on striatal-projection pathways.

Gorelova, N. A. and Yang, C. R. (2000). Dopamine d1/d5 receptor activation modulates a persistent sodium current in rat prefrontal cortical neurons in vitro. *Journal of Neurophysiology*, 84.

Hazy, T. E., Frank, M. J., and O'Reilly, R. C. (2007). Towards an executive without

a homunculus: Computational models of the prefrontal cortex/basal ganglia system. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362.

Jaeger, D., Kita, H., and Wilson, C. J. (1994). Surround inhibition among projection neurons is weak or nonexistent in the rat neostriatum. *Journal of Neurophysiology*, 72.

Jaskir, A. and Frank, M. J. (2023). On the normative advantages of dopamine and striatal opponency for learning and choice. *eLife*, 12.

Lisman, J. E., Fellous, J. M., and Wang, X. J. (1998). A role for nmda-receptor channels in working memory. *Nature Neuroscience*, 1.

Nassar, M. R., Helmers, J. C., and Frank, M. J. (2018). Chunking as a rational strategy for lossy data compression in visual working memory. *Psychological Review*, 125.

O'Reilly, R. C. and Frank, M. J. (2006). Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Computation*, 18.

O'Reilly, R. C. and Munakata, Y. (2000). *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain.* The MIT Press.

O'Reilly, R. C. and Munakata, Y. (2019). *Computational Explorations in Cognitive Neuroscience.*

Randall C. O'Reilly, Yuko Munakata, M. J. F. T. E. H. and Contributors (2012). *Computational Cognitive Neuroscience.* WikiBook.

Shen, W., Flajolet, M., Greengard, P., and Surmeier, D. J. (2008). Dichotomous dopaminergic control of striatal synaptic plasticity. *Science*, 321.

Urakubo, H., Honda, M., Froemke, R. C., and Kuroda, S. (2008). Requirement of an allosteric kinetics of nmda receptors for spike timing-dependent plasticity. *Journal of Neuroscience*, 28.

Wiecki, T. V., Riedinger, K., Ameln-Mayerhofer, A. V., Schmidt, W. J., and Frank, M. J.

(2009). A neurocomputational account of catalepsy sensitization induced by d2 receptor blockade in rats: Context dependency, extinction, and renewal. *Psychopharmacology*, 204.

# CHAPTER 2

# Transformer Mechanisms Mimic Frontostriatal Gating Operations When Trained on Human Working Memory Tasks

Paper in preparation for submission to ICLR 2025

Aneri Soni*[34], Aaron Traylor*[1], Jack Merullo[1], Michael J. Frank†[23] & Ellie Pavlick†[13]

∗ Co-first author

† Co-Senior Author, Equal Contribution

[1] Department of Computer Science: Brown University, Providence, Rhode Island

[2] Department of Cognitive and Psychological Sciences: Brown University, Providence, Rhode Island

[3] Carney Institute for Brain Science: Brown University, Providence, Rhode Island

[4] Department of Neuroscience: Brown University, Providence, Rhode Island

{aneri_soni, aaron_traylor, jack_merullo, michael_frank, ellie_pavlick}@brown.edu

**Abstract**

The Transformer neural network architecture has seen success on a wide variety of tasks that appear to require complex "cognitive branching" – the ability to maintain pursuit of one goal while accomplishing others. In cognitive neuroscience, success on such tasks is thought to rely on sophisticated frontostriatal mechanisms for selective *gating*, which enable role-addressable updating– and later readout– of information to and from distinct "addresses" of memory, in the form of clusters of neurons. However, Transformer models have no such mechanisms intentionally built-in. It is thus an open question how Transformers solve such tasks, and whether the mechanisms that emerge to help them to do so resemble the gating mechanisms in the human brain. In this work, we analyze the mechanisms that emerge within a vanilla attention-only Transformer trained on a task explicitly designed to place demands on working memory gating in computational cognitive neuroscience. We find that the self-attention mechanism within the Transformer develops input and output gating mechanisms, particularly when task demands require them. These gating mechanisms mirror those incorporated into earlier biologically-inspired architectures and mimic those in human studies. When learned effectively, these gating strategies support enhanced generalization and increase the models' effective capacity to store and access multiple items in memory. Despite not having memory limits, we also find that storing and accessing multiple items requires an efficient gating policy, resembling the constraints found in frontostriatal models. These results suggest opportunities for future research on computational similarities between modern AI architectures and models of the human brain.

**Keywords:** transformers; neural networks; working memory; computational neuroscience; gating; computational cognitive science; mechanistic interpretability

## 2.1 Introduction

Computational models based on the Transformer architecture (Vaswani et al., 2017) have seen success on a wide variety of tasks that appear to require complex "cognitive branching": the ability to maintain pursuit of one over-arching goal while performing other subtasks along the way. For example, Transformer-based large language models (LLMs) have demonstrated impressive abilities in not just language (Brown et al., 2020), but planning (Huang et al., 2022), navigation (Du et al., 2021), and problem solving (Lewkowycz et al., 2022).

In humans, there is strong evidence that performance on such tasks depends on a neural mechanism for *gating* (Frank and Badre, 2012; Badre and Frank, 2012; Chatham et al., 2014; Rac-Lubashevsky and Kessler, 2016; Rac-Lubashevsky and Frank, 2021), which manages 1) if new information goes into working memory, 2) where (which "address") that information will be stored, 3) what other already stored information should continue to be robustly maintained, and 4) from which address information will be retrieved to respond to a query. These demands collectively comprise role-addressable input and output gating mechanisms.. Typical Transformer models have no specialized architecture for working memory, in spite of their ability to succeed at tasks which appear to require it. Although some Transformer models have additional built-in structure for memory (Dai et al., 2019; Burtsev et al., 2020), recurrence (Dai et al., 2019), or hierarchy Wang et al. (2019), vanilla Transformer models without any such inductive biases remain the dominant architecture for modern AI systems (Brown et al., 2020; Touvron et al., 2023). This raises the question: in solving such tasks, does a mechanism for selective input and output gating emerge within the vanilla Transformer?

Transformers are good candidates for learning gating behavior because of inductive biases within the self-attention mechanism, i.e., the Transformer's defining architectural component. Within self-attention, numerous "attention heads" construct contextual representations for each item in their input sequence through a learned weighted combination

of the previous items in the sequence. Attention heads could in principle learn gating behavior by marking sequence elements with a key (input gating) and reading out those values later by querying for those keys (output gating). Moreover, the attention mechanism in Transformers is decomposed in a way that enables it to readily differentiate "reading" and "writing" operations. This behavior is analogous functionally to the neurobiological roles of corticostriatal circuits in humans and other animals, in which isolated clusters of prefrontal neurons represent distinct "addresses" in memory that can be updated or read out from via selective gating actions triggered by basal ganglia and thalamus (O'Reilly and Frank, 2006a; Frank and Badre, 2012; Calderon et al., 2022; Soni and Frank, 2024). In computational neuroscience models of this process, the prefrontal clusters (or "stripes") can also serve as latent abstract "roles" that condition how to interpret content within them. When learning effective gating policies, these models afford functions such as variable binding and indirection that support rapid generalization to new situations (O'Reilly and Frank, 2006a; Frank and Badre, 2012; Collins and Frank, 2013; Kriete et al., 2013; Bhandari and Badre, 2018). We hypothesized that if Transformers use their attention heads to learn effective gating strategies, these would similarly facilitate rapid learning and generalization in working memory tasks.

A secondary hypothesis was that Transformers would exhibit similar limitations in effective working memory capacity to those in humans and biological networks. In particular, capacity limits in frontostriatal gating networks do not reflect a limitation in the number of available neural populations for storage, but rather result from a credit assignment problem that arises when learning to manage (store and access) multiple items in memory (Soni and Frank, 2024; Todd et al., 2009). Moreover, these limitations can be partially mitigated by learning to reuse effective gating policies (Soni and Frank, 2024). These results align with human studies showing that working memory capacity is not limited by the number of items one can maintain but rather by their effective gating strategies in frontostriatal circuits (Vogel et al., 2005; McNab and Klingberg, 2008;

Baier et al., 2010). We hypothesized that Transformers would exhibit similar effective limitations on their capacity, which can similarly be mitigated by effective gating policies. Notably, unlike biological networks, Transformers have no inherent working memory limit, because all information is provided within a single input context window. Thus this hypothesis provides a strong test of the computational demand on credit assignment for limiting working memory.

In this work, we train vanilla Transformer models with self-attention on a working memory task paradigm that was specifically designed to evaluate models of selective gating and working memory in computational neuroscience (O'Reilly and Frank, 2006a; Rac-Lubashevsky and Frank, 2021). We use recent techniques from *mechanistic interpretability* (Olah, 2022; Nanda and Bloom, 2022) to expose the mechanism that the Transformer uses in order to perform the task. We find that, as a result of training, the self-attention mechanism specializes in a way that resembles existing models of input-output gating. Specifically, we find that the trained model uses the *key vectors* within the attention mechanism to control *input gating*, i.e., determining which elements in the input to consider vs. ignore, as well as controlling how the information is stored– in other words, assigning it a *role* such that the model can access it later. The model uses the *query vectors* to control *output gating*, i.e., determining which information is accessed in order to complete a task. These mechanisms only arise when the training task places demands on role-addressable gating that mimic those in humans and biological networks. Moreover, when they do arise, they support rapid generalization and improve effective working memory capacity. Our findings highlight the importance of considering the emergent mechanisms that result from training in addition to the innate architectural mechanisms when drawing comparisons between AI systems and human cognitive processes, and opens the door for future analysis and work which can enable more principled studies of the similarities and differences between human vs. machine cognition.

## 2.2 Background

### 2.2.1 Gating Mechanisms

There is strong evidence that working memory in human brains makes use of a *gating mechanism*, which processes and stores information analogously to gates being opened and closed (Rac-Lubashevsky and Kessler, 2016; Rac-Lubashevsky and Frank, 2021; Bhandari and Badre, 2018). The input gate controls which information is stored or not stored in memory, and if stored, into which "address" (population of prefrontal neurons). The output gate controls which content within working memory is accessed in order to produce a response or to make subsequent gating operations. Gating policies are also dependent on the learned task-dependent context (i.e. *role*) of the information to be stored and accessed in working memory.

In cognitive neuroscience, a variety of tasks are used to study the capacity of working memory. In this work, we focus on a variant of the "reference-back 2" task (Rac-Lubashevsky and Frank, 2021). This human experimental task was inspired from analogous designs developed to study selective gating of independent content in frontostriatal neural networks (O'Reilly and Frank, 2006a) and which was used in Chapter 1 (O'Reilly and Frank, 2006b; Soni and Frank, 2024). In the reference-back paradigm, symbols such as letters or numbers are viewed one at a time, and the participant must determine whether the current symbol is the same or different as that stored in memory for a given role (letter or number). This type of memory is role-addressable because it requires comparing the new symbol specifically with the stored item in memory that matches its role (letter or number), which may have been stored in the last trial or several trials back. Participants are also given a cue to indicate whether to update the current symbol as the "reference" to be compared on subsequent trials of the same role, or if instead they should only compare the symbol to the previous reference and continue to maintain that reference for subsequent trials. Thus this task requires selective updating and accessing of information

in a role-addressable manner.

### 2.2.2 Transformers

Transformers are powerful language models which create contextualized representations of the input sequences. They learn to predict the next token one at a time using an "attention" mechanism that scans other tokens in the sequence for relevant information(Bahdanau et al., 2014). A common practice (that is used here) is to mask any future tokens and so predictions and representations must be made by the current token and any previously seen tokens. These models are able to learn and represent complex sequence modelling tasks.

For a given prediction, Transformer attention generates three separate vectors for each token in the sequence: a query, key, and value ($q$, $k$, $v$). The key vector is a set of tokens that the model has learned are most relevant to the token at hand. The query vector scans the tokens in the key vectors and calculates how much the current prediction should "attend" to each of those tokens. Then, the value vectors at those positions are multiplied by the corresponding weights, summed up, and added to the next representation: for token $i$ at layer $j$, the contextual representation is $\sum_k q_i^j \cdot k_k^j * v_k^j$. Thus, the next token prediction includes earlier sequential information by combining the value vectors from previous tokens. In other words, Transformer attention can be viewed as a read/write mechanism: for a given token, the queries and keys dictate which tokens to read from, the values are the content that is read (proportional to the attention calculated by the keys and queries), and the summed content is written to a new representation at the given token. As we shall see below, the comparison to role-addressable input and output gating operations is evident. The key vectors form addresses analogous to the PFC "stripes" (neural populations that support variable binding in memory). The learned key construction determines the address to store an item and is thus analogous to input gating. Conversely, the query vectors determine which addresses are accessed, and are

thus analogous to output gating.

**Limitations as Cognitive Models:** There are some significant differences between how Transformers and humans solve tasks. In particular, because Transformers can attend to any part of the sequence when creating a representation, they are not limited by memory constraints. Transformers can solve tasks that would push the limits of human working memory(but it should be noted that Transformers require disproportionately large amounts of training data to do so). Nevertheless, we hypothesized that Transformers might still learn effective gating policies that mimic those in frontostriatal networks. Moreover, as briefly reviewed in the introduction, working memory capacity in humans and biological computational models is not limited primarily by the memory demand *per se*, but rather by the difficulty of the credit assignment problem for learning how to manage role-addressable storage and access of multiple items in memory. A fundamental bridge between Transformers and other models of WM (and humans themselves) would be if **Transformers also needed to overcome the credit assignment learning problem, despite an unlimited memory capacity**.

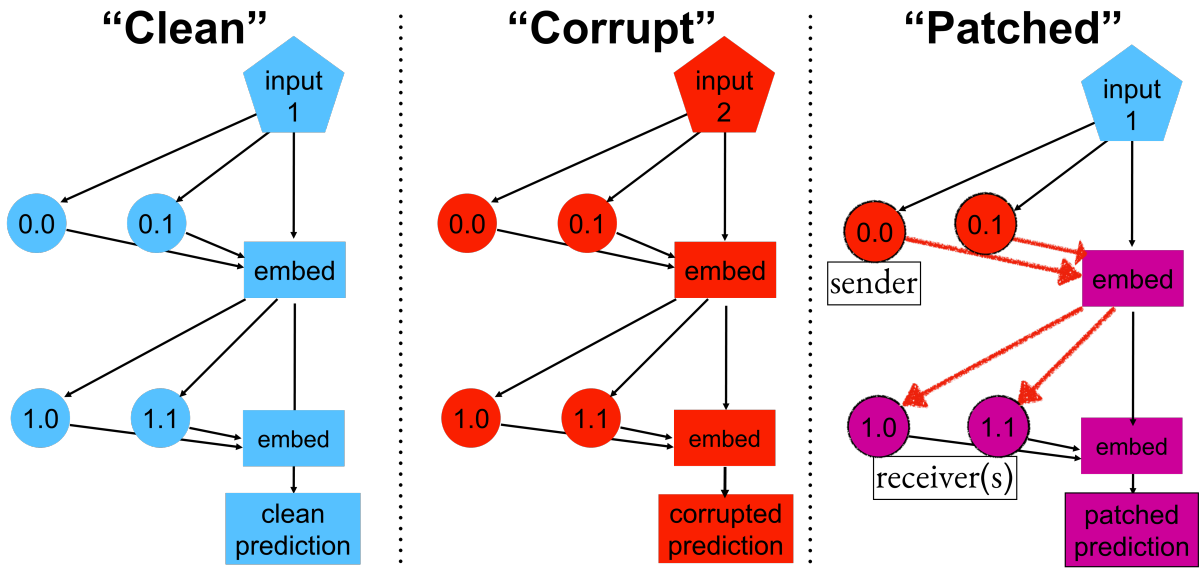



Figure 2.1: **Graphical Path Patching** Graphical diagram of the path-patching process. Attention heads are represented as circles (layer,head index), and contextual representations of each token (as well as the next token prediction) are represented as rectangles.

### 2.2.3 Mechanistic Interpretability

We use a set of recently introduced analysis tools (Elhage et al., 2021) which enable us to uncover specific mechanisms defined in terms of model weights within the Transformer. Specifically, we use path-patching (Wang et al., 2022; Goldowsky-Dill et al., 2023), a generalization of causal mediation analysis (Pearl, 2001) that allows us to determine which components of a neural model (e.g., attention heads) work together in order to produce observed behavior on a task. The discovered components are referred to as a *circuit* (Räuker et al., 2023).

Path-patching involves making a incisive edit to the representations of a trained model and observing how the model's behavior is affected, allowing one to infer the computations implemented within individual attentional heads (see Fig. 2.7). Path-patching typically requires a minimal pair of examples: the "clean" example and the "corrupted" example, in which one token from the clean example is changed, as well as the correct label. Given representations from the model for both the clean and the corrupted examples (the blue and orange components in the figure), we can chose a specific component anywhere in the model (referred to as the "sender"), and replace the clean representation with the corrupted one at that specific component. These embeddings will be received by the next layer ("receiver"), thereby "patching" the path. From there, the patched model will compute the new prediction. In the figure, we send from layer 0, head 0 and 1 to layer 1, both heads 0 and 1. All clean representations that are not along this path are not modified and are unaffected by the patch. The model then recomputes all representations after the receiver (the "patched" representations), and arrives at a new prediction. If the model output matches the corrupt prediction rather than the clean one, that prediction is causally dependent on the path from sender to receiver. See (Wang et al., 2022) and (Goldowsky-Dill et al., 2023) for a more comprehensive review of path-patching methods.

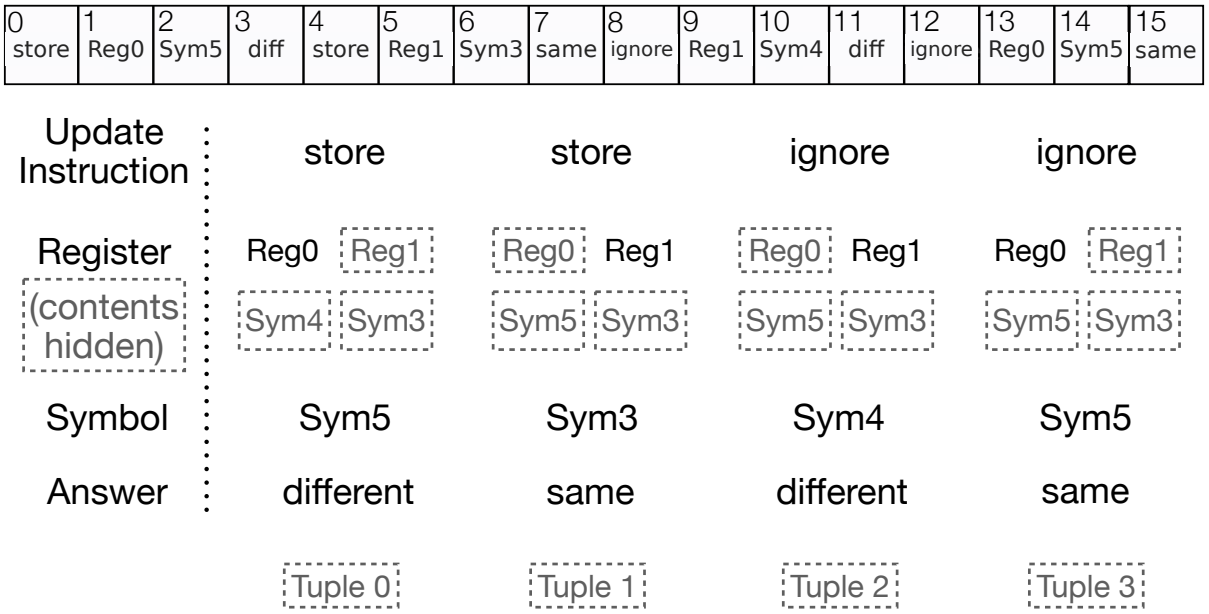



Figure 2.2: **Task** Above: example of textual reference-back task as model input. Below: step-by-step task process; models do not view task-internal grey words. "Update Instruction" executes after "Answer" despite appearing earlier sequentially.

## 2.3 Task

We create a modified text-based version of the reference-back 2 task (Rac-Lubashevsky and Frank, 2021) designed to tax selective WM gating (O'Reilly and Frank, 2006a).The *textual reference-back task* requires making same/different judgments between incoming symbols assigned to a particular "register" in memory, with respect to those seen previously and linked to those same registers. Like the original tasks, the textual reference-back task is sequential, and requires independent updating and maintenance of two memory registers, each containing one of $S$ arbitrary *symbols* at a time. At the beginning of each sequence, each register is initialized individually to one $s \in S$ (the pool of symbols is shared between registers, which was shown to more substantively tax gating mechanisms in (O'Reilly and Frank, 2006b).) Each sequence is composed of $L$ tuples, each containing register address $\text{Reg}_i$, symbol $\text{Sym}_i$, same/different label $\text{Ans}_i$, and update instruction $\text{Ins}_i$. For a tuple $i \in L$, the **answer** $\text{Ans}_i$ is a binary value that is either Same if symbol $\text{Sym}_i$ is currently stored in the register with address $\text{Reg}_i$, or Different otherwise. The **update**

**instruction** $\text{Ins}_i$ also takes one of two values, evenly distributed:`ignore` or `store`. If the instruction is `ignore`, then the model still needs to make the same/different determination with respect to the stored reference, but the new symbol should not update the register content (i.e., the reference remains unperturbed). If it is `store`, then from that point on in the sequence, $\text{Sym}_i$ is stored in the register with address $\text{Reg}_i$ until otherwise updated. An example is shown in Fig. 2.2.

We implement each reference-back task example in our data as a single sequence, and measure models' ability to predict Same versus Different for each $\text{Ans}_i$. Each sequence has 10 same/different answers, and we generate 100,000 train, 1,000 validation, and 1,000 held-out test sequences.

The class balance of `same` to `different` answer labels in the train/test datasets is roughly 1:2, making a "maximum class" heuristic solution 0.66 accuracy, 0.33 precision, and 0.5 recall. We test several other heuristics, the strongest of which is predicting `same` if another tuple including `Store` and the target register and target symbol exists in the sequence, which scores 0.80 accuracy, 0.82 precision, and 0.85 recall.

The computational advantages of role-addressable gating in frontostriatal networks (relative to other recurrent neural networks) are particularly evident when any symbol can be assigned to any register, so that the networks have to learn to assign them separable addresses O'Reilly and Frank (2006a). To test if similar constraints impact learned gating strategies in Transformers, we included additional controls. The "split set" control made a mutually exclusive list of symbols for each register and a separate symbol list if the update instruction was ignore. The split set control (ignore integrated) was when each register had a mutually exclusive group of symbols but the ignore update instruction sampled from the corresponding register that was being updated. These controls greatly simplify the task, as the network no longer needs to manage interference that could arise when performing same/different operations across registers that could have stored the same symbol. We hypothesized that the Transformer would only learn effective gating

policies when task demands required them, and moreover that these would facilitate generalization. To evaluate this latter claim, in a series of held out experiments, we held out certain percent of the symbols from one of the two registers, and tested the model's ability to generalize to that symbol register combination.

Finally, we taxed the networks effective capacity by increasing the number of registers from 2 to 3. We hypothesized that learning this task would be disproportionately more difficult given the need to learn to manage storage and access of multiple registers, but that performance would be facilitated in networks that had learned effective gating policies.

## 2.4 Model

We use vanilla Transformers to facilitate interpretability, as in prior work that analyzes emergent mechanisms (Elhage et al., 2022). Our models contain two decoder-only layers, each with only two heads (four in total), and no multilayer perceptrons or layer normalization, followed by a linear "unembed" layer to project the output of the last decoder into the space of the entire vocabulary at each timestep [1]. Our network uses absolute positional embeddings (Vaswani et al., 2017). The vocabulary contains all possible tokens, represented individually with embedding size $E$. Models are trained to predict the next token with the language modelling objective: if the model is predicting $\text{Ans}_c$, it will have access to all ($\text{Ins}_i$, $\text{Reg}_i$, $\text{Sym}_i$, $\text{Ans}_i$) tuples where $i < c$, as well as $\text{Ins}_c$, $\text{Reg}_c$, and $\text{Sym}_c$. However, the models only receive loss at positions where a same/different token must be predicted (this is similar to the reward function applied in frontostriatal gating networks; O'Reilly and Frank (2006a); Soni and Frank (2024)). Furthermore, each layer gets a causal attention mask– when constructing each token representation, it cannot look ahead at tokens further down the sequence.

The models are trained over 60 epochs of the 100k training data points, learning from

[1] In practice, only 'same' and 'different' are ever predicted.

6 million examples in total. Models are evaluated on their accuracy (whether the correct $\text{Ans}_i$ is predicted for each tuple $i$), measured in precision and recall, as well as the `same` versus `different` token logit difference.

## 2.5 Experiments

First, we select and analyze a single Transformer model which succeeds on the task, and upon investigation of its weights find that it learns a mechanism for input/output gating. Second, we conduct a search over more trained models, wand find that model performance correlates with markers of learning a gating policy, analogous to findings in the frontostriatal neural networks (Frank and Badre, 2012; Soni and Frank, 2024).

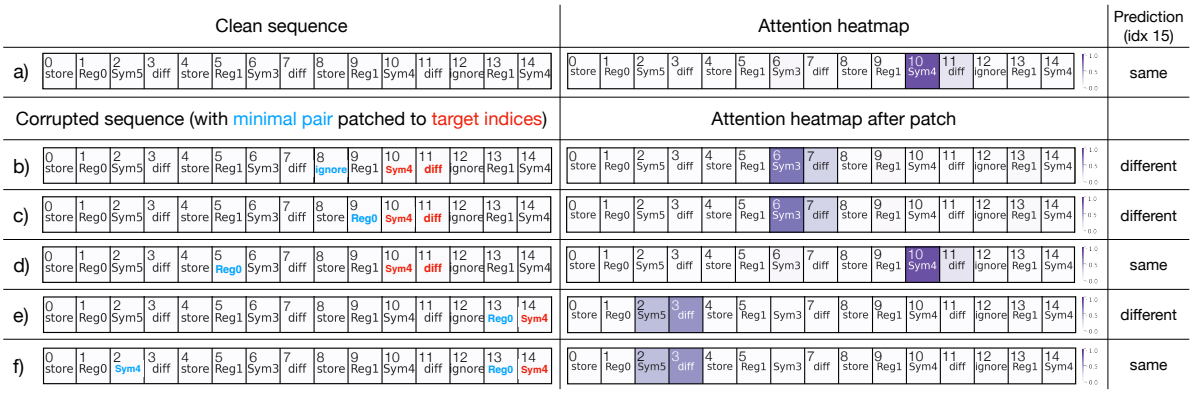



Figure 2.3: **Path Patching Examples.** Model behavior when predicting same/different (token 15) is shown. We measure attention visualized as a shade of purple, with deeper shade corresponding to higher attention to that token. We create "corrupted" minimal pairs in which changing a token (light blue) either changes the correct label at index 15 (examples b, c, e) or does not (d, f). We make small path-patching edits with the minimal pair to targeted network components (layer 1 keys for b, c, d, f; queries for e,f). In other words, we replace specific components (denoted with red text) with their corresponding representation from the "corrupted" sequence, but hold all other representations constant, and then obtain a new same/different prediction. In all test examples, making the small patch successfully alters the model's prediction to align with the "corrupted" example, providing causal evidence for this network component is responsible for the gating operation.

We first establish that a Transformer model is able to succeed on the reference-back-2 task. We perform a small hyperparameter search, details of which are included in the

Appendix. and select a model that reaches 100% accuracy on the test data for further analysis. We determine the circuit that the model uses through an array of path-patching experiments with a simple minimal pairs paradigm. Our "sender" within path-patching is always both attention heads at layer 0, and our "receiver" is always both attention heads at layer 1.

At layer 0, the model learns to condense the task-critical information from each tuple into one embedding, at the position for $\text{Sym}_i$[2]. At this layer, the model pays 85.8% of total attention to the task-critical information to that tuple, and just 14.2% of attention to other tuples.

At layer 1, the attention heads learn to attend to the $\text{Sym}_i$ key vector representing the tuple where information was last stored in the target register. The heads pay 70.2% of total attention to this tuple (the "stored" tuple), and only 29.8% of attention to all other tokens. This behavior is tied to the target register matching the register in the stored tuple, which is analogous to gating of the relevant role-addressable PFC stripe (O'Reilly and Frank, 2006a; Soni and Frank, 2024). We focus our analysis on the Layer 1 representations which exhibit this learned gating policy, shown in Fig. 2.3.

### 2.5.1 Input Gating through Key Vectors

Input gates in working memory control what incoming information is remembered and are "role-addressable"– i.e., stored in memory in such a way that it can be freely accessed later when it is needed for task completion. In a Transformer, we show below that the key vectors serve the role of addresses (analogous to PFC "stripes"). These values are retrieved based on their match to a query vector at a current or later time during self-attention. Thus the input gating in Transformers is controlled through key construction: the composition of the output of the Layer 0 attention controls which

[2]Redundantly, the model does the same at the position for $\text{Ans}_i$. Through additional experimentation, we determine that this is a quirk of Transformer learning, and does not impact our analysis.

content is stored in the key vector at Layer 1 for later use. At a later timestep, a query vector will address the information in the key vector.

In our model analysis, we find that key composition at the $\text{Sym}_i$ position (positions 2, 6, 10, and 14 in Fig. 2.3a) roughly represents each tuple. A query's ability to address this position depends on whether the represented tuple contains a `Store` or an `Ignore`. Key vectors representing an `Ignore` tuple receive very little attention (0.4% of layer 1 attention averaged over test set), whereas those representing a `Store` tuple receive the bulk of the attention (86.8%). We determine this effect causally with path-patching (Fig. 2.7). First, we create clean sequences sampled from our test set, and then corrupt these sequences by switching a `Store` within tuple $i$ to an `Ignore`. We then path-patch only the key vectors of $i$. We expect, if the key controls input gating, that patching these key vectors should "block" attention to all of tuple $i$. An example attention pattern is in Fig. 2.3, examples a and b. We find that the model's attention shifts away from the tuple accordingly in 100% of patched instances. The presence of an `Ignore` or a `Store` within a tuple controls whether the key construction acts as an open input gate or a closed input gate.

Key construction also depends on the *role* of the represented content; within our task, that means whether $\text{Reg}_i$ is Register 0 or Register 1. When making a same/different prediction, key vectors representing a tuple that matches the target register receive most of the model's attention (92.5% of total attention), while those that do not match are not attended to (3.3% of total attention). Again, we use path-patching to determine that key construction encodes roles. This time, given a target tuple $i$ with a target register, we corrupt the register of the stored tuple, changing it from Register 0 to Register 1 or vice versa; to predict the answer for $i$, the model must attend to an earlier tuple with the target register. An example is Fig. 2.3, row c. The model's attention shifts away accordingly across every example in the test set. Note that the stored tuple must be modified; if the same corruption is made earlier (as in row d), attention does not shift.

This behavior shows that the gating within self-attention is *role-addressable*; the registers within the task function as roles, and are embedded within the key vectors as part of the representation.

### 2.5.2 Output Gating through Query Vectors

Within working memory, output gates control which addressable information is accessed in order to complete a task. Given that key vectors serve the role of addresses, query vectors in turn control which key vectors are accessed, through the final Q*K dot product in attention. Query construction thus performs the role of output gating within Transformers.

The query composition controls which addressable Symbol i representations are attended to based on the identity of the target register.We determine this through a final set of path-patching experiments, an example of which is shown in Fig. 2.3, row e. Rather than editing the register in the stored tuple as done in row c, we corrupt the target register itself; this means that the query must now find representations corresponding to the other tuple. In row e, the model finds $Sym_5$, and predicts `different`; and in row f, we patch in $Sym_4$ at index 2, and the model predicts `same`.

Upon inspection, the query corresponds with key vectors that represent tuples which also contain the target register. When we edit the target register in minimal pair experiments, we observe that the attention shifts from the original stored tuple (74.1% of attention) to the stored tuple that matches the edited register, and successfully makes an updated same/different comparison to the symbol in the edited register in every instance.

Editing aspects of the target tuple other than target register has minimal effect on the query construction behavior. No edits to the query cause the model to attend to a `Ignore` tuple, further evidencing of output gating behavior– **only content that has been made "addressable"** can be accessed for a response. Furthermore, we find that the target instruction and symbol do not factor into the query composition– changing them through path-patching to the query does not affect attention. This is notable because the model

could employ other strategies for determining which tuples are eligible to be the stored tuple; e.g. attending to all symbols to match if any of them are the same as the target symbol.

### 2.5.3 Successful Task Performance is Related to Discovering Gating Policies

Is learning this gating policy useful for succeeding on the task? To answer this question, we compare models with the same hyperparameters across different random seeds. After measuring model performance on the target reference back-2 task and assessing the gating policy (accuracy on the input and output gating subtasks), we find stark qualitative and quantitative differences between models that learn and don't learn the gating policy.

We train 20 new models, each with a different random initialization, and measure both training loss and test set accuracy. 5 of the models succeed 100% of the time, and the other 15 models succeed between 94%-99.99% of the time, with a mean of 97.72% and a standard deviation of 2.03.

We observed in the prior sections that the trained Transformer model uses its key and query composition to control role-addressable input and output gating, respectively. To identify whether the new models learn to gate similarly, we evaluate the key and query composition of all 20 new models by making minimal pair path-patching edits for every test example, where the answer changes from same to different or vice versa, similarly to Figure 2.3.

To evaluate the ability to open and close input gates, we corrupt the stored tuple's `Store` to `Ignore`, and path-patch only to the stored tuple's *key vectors*. To evaluate the role addressability of the content during output gating, we corrupt the target register (changing `Reg0` to `Reg1` or vice versa), and path-patch only to the *query vector* for making the same/different judgment.

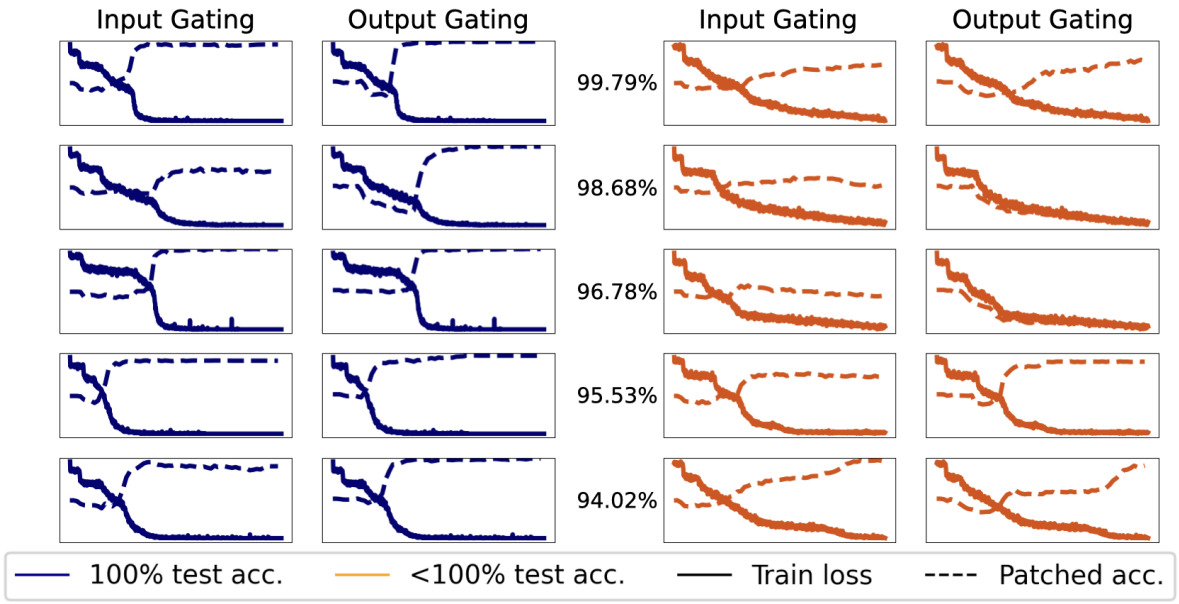



Figure 2.4: **Model Performance** over training on patching subtasks. Each graph contains an individual model's training loss (solid line) and subtask accuracy (dashed line, between 0 and 1) over time; the line's color corresponds to whether the model reaches 100% accuracy on the general test set.

We visualize the 5 runs that reach 100% accuracy on the test data in Figure 2.4, as well as 5 randomly selected runs that do not reach 100% accuracy on the test data, and plot their training loss versus their accuracy (between 0 and 100) on the two patched subtasks over the course of training.

Two trends become apparent from the data: first, models which score a perfect test accuracy appear to succeed at the subtasks more readily than models which do not. Of the former, 3 of 5 models reach 100% accuracy on both subtasks readily, plateauing less than halfway through training. In contrast, models that make errors in the test set also do not reach such immediate success at the patched subtasks (including the 10 not pictured in this graph); in fact, many categorically fail, scoring as low as 49% accuracy. These results do not indicate that this class of models' representations are useless for the task– they all score between 94% and 99.99%, well above heuristic performance. We take these results as evidence that learning a robust policy for gating correlates with model performance at the textual reference-back task.

The second emergent trend is that many models across both classes have a sharp decline in training loss, which correlates with a similarly steep increase in accuracy on both subtasks. Sudden jumps in performance is a noted phenomenon that has been observed in other Transformer models in cases of e.g., grokking (Power et al., 2022). We interpret this phase transition as suddenly learning a gating mechanism. Models that do not exhibit phase transitions to the same degree take longer to fit the task, and do not reach high subtask accuracy.

### 2.5.4 Emergent Gating Policies Due to Task Demands

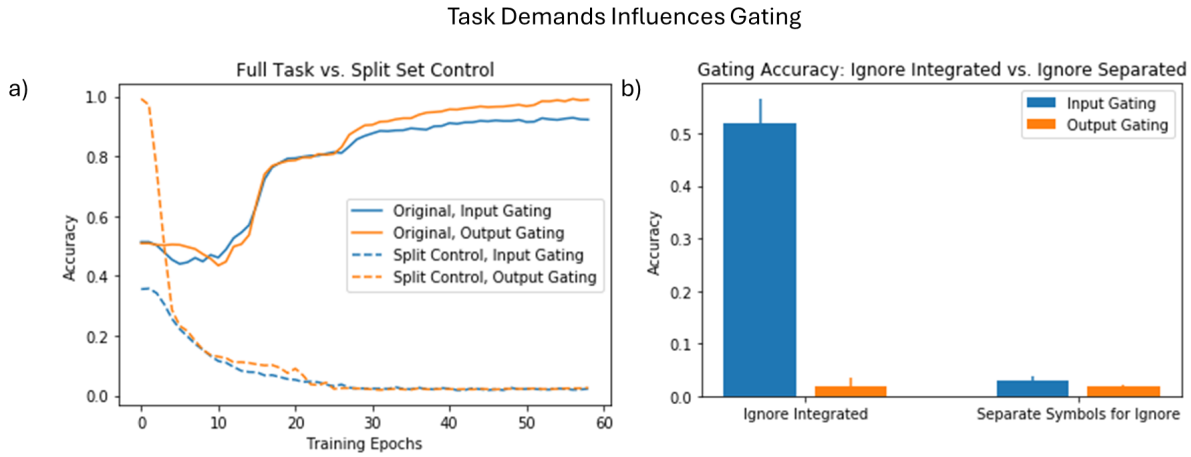



Figure 2.5: **Split Set Control** a) Ignore Symbols are not mutually exclusive with the registers. The models must learn to respond appropriately to store/ignore instructions (input gating). This control task separates the symbols shown to each register, reducing the need for the model to learn output gating. b) Both registers and the Ignore instruction have mutually exclusive symbols.

Under which conditions to gating computations arise? One possibility is that they are a trivial consequence of the Transformer's architecture. After all, gating is sometimes expressed as a simple multiplicative operation, as in LSTMs (Hochreiter and Schmidhuber, 1997). And indeed, the key, query, and value vectors underlying the attention heads are combined via matrix multiplications. However, this alone does not ensure that the networks learn effective, role-addressable gating policies. Indeed, we have already shown that with some random initializations, some networks can perform quite well (but not

100%) without learning gating policies. We next sought to establish the training task demands that induce such gating policies. To do so, we ran a "split set" control experiment which simplifies the task, removing the role addressability of gating requirements. Using the same 50 symbols, each register is associated with a mutually exclusive subset of symbols. Symbols 0 to 16 are always trained with Register 1, and Symbols 16 to 33 are trained with Register 2. For Ignore update instructions, the Symbols are sampled from Symbols 34 to 50. Therefore the model does not need to learn to differentiate between store and ignore in a meaningful way and does not need to assign specific addresses for key and query construction to specific registers. Indeed, we previously showed that the advantage of frontostriatal gating networks relative to other recurrent networks is largely reduced in such scenarios (O'Reilly and Frank, 2006a). We thus hypothesized that while this task greatly facilitates learning, the Transformer will learn an overly memorized, brittle solution and will not develop effective input and output gating strategies. Indeed, networks trained on the split set control task do not perform well at either the input or the output gating subtasks. To dissect the problem further, we do a second control, referred to as split set, ignore integrated. This time each register sees 25 symbols (total of 50 symbols). Under an `Ignore` instruction, the symbol selected will be in distribution with the chosen register. In other words, the model must distinguish between a `Store` and `Ignore` instruction and thus learn input gating. However, it still does not have to learn an output gating policy because both registers have mutually exclusive symbols. These models indeed show that they perform well at the Store vs Ignore input gating tasks, but not the output gating task 2.5b.

### 2.5.5 Effective Gating Facilitates Generalization to Out-of-Distribution Symbol-Register Pairs

Learning a more general gating strategy should allow models to generalize to out of distribution examples. To assess this, we held out a subset of symbols from each register

(randomized which symbol would be held out from which register). We tested these models on a challenge dataset which included examples of in-distribution pairings (register-symbol pairings that were seen in training) and out-of-distribution pairings (register-symbol pairings that were not seen during training). When holding out 5% of the symbols, the models learn a robust input and output gating strategy, and notably, perform on average at 99.7% for out-of-distribution symbol-register pairings. In contrast, the models trained on the split set controls do not learn robust gating strategies (despite performing perfectly at the trained task) and accordingly perform more poorly on the out of distribution examples: 77.2% (ignore integrated) and 88.6% (ignore separated).[3]

Note that the Split Set (Ignore Separated) task depends least on learning an effective gating strategy. These models learn the task very quickly and show no ability to perform on the gating subtask, and instead likely learn heuristic solutions. Interestingly, when tested on the challenge dataset (which includes examples of both in distribution and out of distribution register-symbol pairs), these models show a large disruption in their ability to perform on in distribution sequences (80.8%). This insinuates that the strategy learned by these models is highly dependent on memorization and by adding in new elements, the strategy fails. The out of distribution accuracy is slightly higher than the in distribution accuracy and future work could try to better understand the effects of perturbing a brittle model.

### 2.5.6 Increased Task Demands: 3 Registers

Thus far, the experiments have focused on tasks with 2 registers, mimicking that used in the reference back-2 task (Rac-Lubashevsky and Frank, 2021). However, human

[3]One concern is that in the 5% held out, each register is trained on 49 symbols while in the split set (ignore integrated), each register is trained on 25 symbols. While all other parameters are held constant, this difference might be big enough to account for the large difference in generalization. To account for this, we ran another set of simulations holding out 60% of the symbols (each register is trained on 35 symbols). These models robustly learn a gating policy. Even with a drastic increase in the number of held out symbols from 5% to 60%, the out-of-distribution accuracy is still very high compared to either of the split set controls.

| Experiment | Out of Distribution | In Distribution |
|---|---|---|
| 5% Held Out | 99.7% | 99.8% |
| 60% Held Out | 95.3% | 99.3% |
| Split Set (Ignore Integrated) | 77.2% | 94.6% |
| Split Set (Ignore Separated) | 88.6% | 80.8% |

Table 2.1: **Accuracy for In Distribution and Out of Distribution Symbol-Register Pairings in Challenge Dataset** Experiments and their associated performance on the challenge data set. The challenge dataset includes a mix of in distribution and out of distribution symbol-register pairings. This table breaks down the accuracy based on if the symbol-register pairing was in the training (in Distribution) or not seen during training (out of Distribution).

working memory has a capacity of about 3-4 items (Cowan, 2008), albeit with vigorous debates questioning whether this limit is discrete or continuous (Zhang and Luck, 2008; Wei et al., 2012; Luck and Vogel, 2013). More recent models and data suggest a "chunking" hybrid between the two, whereby multiple memoranda can compete for shared continuous resources within discrete slots (Nassar et al., 2018; Soni and Frank, 2024). Chunking increases *effective* capacity, allowing more items to be remembered at the cost of precision of some of the items. Notably, in frontostriatal gating models, when the number of registers to manage was larger than 2, networks given limited memory allocation but with chunking capabilities performed better than those that were allocated as many PFC populations as items to store (Soni and Frank, 2024). The reason for this seeming paradox is **credit assignment**: as the number of PFC populations (stripes) increases, the gating management problem becomes more challenging – the network has to learn to route each item to distinct populations and to also learn to read out from the corresponding population for a given probe. Moreover, these learning problems are interdependent.

Because these limitations are computational rather than anatomical (i.e., not driven by the number of neurons/populations available), we hypothesize that they would also manifest in Transformers, even though they have no inherent memory demands at all (since all information is available in the context window). Specifically because of the flexibility and expressivity of Transformers, we predicted that with increasing task demand,

the network would learn the task but would much less robustly learn effective gating strategies that facilitate generalization.

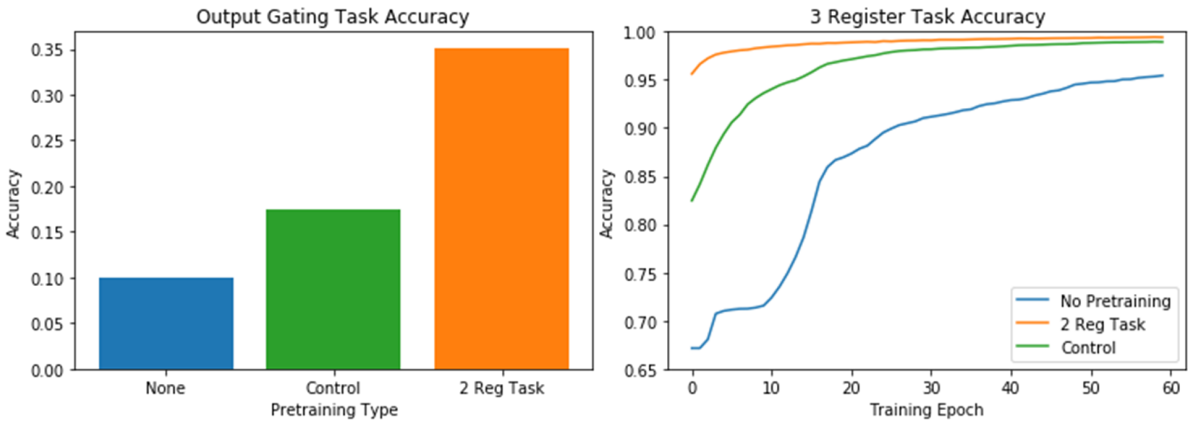



Figure 2.6: **3 Register Task and Pretraining** a)At the end of training on the 3 Register Task, models with 2 Register Task pretraining perform the best on output gating tasks, insinuating learning of a mechanistic solution. b) Accuracy curves through training (after the pretraining) show a stark difference between pretraining on the normal 2 register task and the control task. Models pretrained on the 2 register task, generalize quickly (first few epochs) and with high accuracy (above 95%) on this new task. At the end of training, no pretraining models are below 95% accuracy - which is below the accuracy that 2 register task pretraining models showed at the beginning of training. Control pretraining models show some generalization (due to learning of basic elements e.g. symbols) but do not generalize to the same degree. Models pretrained with the 2 register tasks, which should encourage a mechanistic solution, perform well at the new 3 register task, showing a superior ability to generalize. 22 random seeds were run for each experimental condition and the results here an average of those models. See Appendix for comparison on input and output gating task accuracies.

To test this hypothesis, we increased the number of registers from 2 to 3. First, we confirmed that asymptotic accuracy dropped to 95.4%. This result is qualitatively the same even when training for twice the number of epochs. This is non-trivial given that we are just adding 1 register. We predicted that the degree to which the transformer solves the task is related to heuristics and memorization rather than gating in these cases. We predicted that if we first pretrained the model to effectively manage 2 registers, the network will be able to scaffold the learned gating strategy to learn the three register task more robustly. We further predicted that this pretraining would only be useful if pretraining encouraged gating strategies. Results supported both of these conclusions

Figure 2.6. Networks that were pretrained with 2 registers showed very rapid learning in the 3 register task in the first few epochs. Moreover, this pretraining was far less effective when it did not encourage gating (the split symbol control from above) 2.6b. Finally, we further confirmed that pretrained networks on the original reference back-2 task exhibited higher gating sub-task accuracy 2.6a.

## 2.6 Summary and Discussion

In this work, we investigate Transformer models for emergence of a learned *gating mechanism*; a network component performing role-addressable gating, similar to that in working memory of humans. We observe that the model learns a gating policy and find that task performance is correlated with gating ability. Our results show how learning gating mechanisms is one way Transformers can excel at cognitive branching tasks.

The Transformer models are capable of making use of key composition for input gating and query composition for output gating on the task. We find that making precise corruptions to specific architectural elements of the network causes the model's prediction to change from *Same* to *Different* or vice versa, indicating that those components are causally responsible for the gating mechanism. The architectural biases of attention within the vanilla Transformer model lend themselves well to representing role-addressable content. The learnable nature of keys, queries, and values allows the model to learn to create internal representations. These representations can be learned to signify roles and addresses, mimicking the variable binding and input / output gating mechanisms in biological neural networks (O'Reilly and Frank, 2006a; Frank and Badre, 2012; Collins and Frank, 2013; Kriete et al., 2013).

When we trained more models on the task, we found that the models which perform best on the task correlate with the markers of gating we observed in our circuit analysis, and that the learning trajectory shows a steep decrease in training loss and a steep rise

in patched subtask accuracy simultaneously, suggesting that the model has learned a component of the gating policy at that time. Both findings are analogous to those of (Frank and Badre, 2012), in which they find that networks which learned a hierarchical gating policy performed better at a hierarchical learning task, and humans that learn this policy also show a sharp decrease in loss when they discover it. There is still more work to be done to better understand the models that don't learn the gating policy: what types of solutions do they learn? How do we push models towards a mechanistic solution by hyperparameter tuning? Understanding grokking phase planes in a similar manner as Liu et al. could be informative.

Nevertheless, we show that a critical factor controlling the learned gating policy is the task demands. We found that experimental conditions in which biological networks exhibit gating advantages were similarly needed to give rise to learned gating policies in Transformers. Conversely, when the task places less demands on role-addressable gating (our split symbol control conditions), the models did not learn gating policies, were less able to generalize to out-of-distribution pairs, and were less able to rapidly acquire tasks with higher gating demands (three registers).

We show that models that learn and solve the tasks using gating mechanisms are better at generalization. Further we show how learning brittle and memorized solutions causes some models to falter at even in distribution pairings when they are mixed with out of distribution examples. These results can be used to inform larger models that are needed for better and faster generalization. Future work can characterize what kinds of mistakes are common within the mechanistic and heuristic models to further characterize these models and promote better generalizability.

An important part of human WM is the learning process. Biological WM models show that unlimited WM does not always lead to better performance because it is not trivial to learn how to manage and utilize resources. This is known as the credit assignment problem. We showed that this computational principle holds in Transformers.

By increasing the task demands by 1 register, we found a clear decrease in performance, with a corresponding decrease in gating strategies. However, when simplying the learning problem (by pretraining with the 2 register task -where gating is learned), models learn quickly and with high accuracy on gating subtasks.

Ultimately, finding connections between emergent behavior of Transformer models and human working memory serves to benefit both computational cognitive neuroscience and artificial intelligence. Although Transformer models themselves are limited in their biological plausibility, in this setting they learned behavior mimicking the functionality of working memory, and their application within computational models of the brain should be further explored. From the perspective of artificial intelligence, understanding the strengths and limitations of Transformer models on cognitive branching tasks may inform model analysis across the many diverse settings in which these models are applied.

## 2.7 Acknowledgements

We would like to thank the members of the LUNAR lab and the Laboratory of Neural Computation and Cognition at Brown University for their thoughtful comments and discussion. This work was supported by ONR MURI Award N00014-23-1-2792.

# References


Badre, D. and Frank, M. J. (2012). Mechanisms of hierarchical reinforcement learning in cortico–striatal circuits 2: Evidence from fmri. *Cerebral cortex*, 22(3):527–536.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Baier, B., Karnath, H. O., Dieterich, M., Birklein, F., Heinze, C., and Müller, N. G. (2010). Keeping memory clear and stable - the contribution of human basal ganglia and prefrontal cortex to working memory. *Journal of Neuroscience*, 30.

Bhandari, A. and Badre, D. (2018). Learning and transfer of working memory gating policies. *Cognition*, 172:89–100.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Burtsev, M. S., Kuratov, Y., Peganov, A., and Sapunov, G. V. (2020). Memory transformer. *arXiv preprint arXiv:2006.11527*.

Calderon, C. B., Verguts, T., and Frank, M. J. (2022). Thunderstruck: The acdc model of flexible sequences and rhythms in recurrent neural circuits. *PLoS Computational Biology*, 18(2):e1009854.

Chatham, C. H., Frank, M. J., and Badre, D. (2014). Corticostriatal output gating during selection from working memory. *Neuron*, 81(4):930–942.

Collins, A. G. and Frank, M. J. (2013). Cognitive control over learning: creating, clustering, and generalizing task-set structure. *Psychological review*, 120(1):190.

Cowan, N. (2008). Chapter 20 what are the differences between long-term, short-term, and working memory?

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Du, H., Yu, X., and Zheng, L. (2021). Vtnet: Visual transformer network for object goal navigation. *arXiv preprint arXiv:2105.09447*.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., and Olah, C. (2022). Toy models of superposition. *Transformer Circuits Thread*. https://transformer-circuits.pub/2022/toy_model/index.html.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., et al. (2021). A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1.

Frank, M. J. and Badre, D. (2012). Mechanisms of hierarchical reinforcement learning in corticostriatal circuits 1: computational analysis. *Cerebral cortex*, 22(3):509–526.

Goldowsky-Dill, N., MacLeod, C., Sato, L., and Arora, A. (2023). Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short term memory. neural computation. *Neural Computation*, 9.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. (2022). Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR.

Kriete, T., Noelle, D. C., Cohen, J. D., and O'Reilly, R. C. (2013). Indirection and symbol-like processing in the prefrontal cortex and basal ganglia. *Proceedings of the National Academy of Sciences*, 110(41):16390–16395.

Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V.,

Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., et al. (2022). Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Liu, Z., Kitouni, O., Nolte, N., Michaud, E. J., Tegmark, M., and Williams, M. (2022). Towards understanding grokking: Aneffective theory of representation learning. *NeurIPs*.

Luck, S. J. and Vogel, E. K. (2013). Visual working memory capacity: From psychophysics and neurobiology to individual differences.

McNab, F. and Klingberg, T. (2008). Prefrontal cortex and basal ganglia control access to working memory. *Nature Neuroscience*, 11.

Nanda, N. and Bloom, J. (2022). Transformerlens. https://github.com/neelnanda-io/TransformerLens.

Nassar, M. R., Helmers, J. C., and Frank, M. J. (2018). Chunking as a rational strategy for lossy data compression in visual working memory. *Psychological Review*, 125.

Olah, C. (2022). Mechanistic interpretability, variables, and the importance of interpretable bases. https://www.transformer-circuits.pub/2022/mech-interp-essay.

O'Reilly, R. C. and Frank, M. J. (2006a). Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. *Neural computation*, 18(2):283–328.

O'Reilly, R. C. and Frank, M. J. (2006b). Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Computation*, 18.

Pearl, J. (2001). Direct and indirect effects. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, page 411–420, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Power, A., Burda, Y., Edwards, H., Babuschkin, I., and Misra, V. (2022). Grokking:

Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*.

Rac-Lubashevsky, R. and Frank, M. J. (2021). Analogous computations in working memory input, output and motor gating: Electrophysiological and computational modeling evidence. *PLoS computational biology*, 17(6):e1008971.

Rac-Lubashevsky, R. and Kessler, Y. (2016). Dissociating working memory updating and automatic updating: The reference-back paradigm. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 42(6):951.

Räuker, T., Ho, A., Casper, S., and Hadfield-Menell, D. (2023). Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 464–483. IEEE.

Soni, A. and Frank, M. J. (2024). Adaptive chunking improves effective working memory capacity in a prefrontal cortex and basal ganglia circuit. *eLife*.

Todd, M. T., Niv, Y., and Cohen, J. D. (2009). Learning to use working memory in partially observable environments through dopaminergic reinforcement. In *Advances in Neural Information Processing Systems 21 - Proceedings of the 2008 Conference*.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Vogel, E. K., McCollough, A. W., and Machizawa, M. G. (2005). Neural measures reveal individual differences in controlling access to working memory. *Nature*, 438.

Wang, K., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. (2022). Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*.

Wang, Y.-S., Lee, H.-Y., and Chen, Y.-N. (2019). Tree transformer: Integrating tree structures into self-attention. *arXiv preprint arXiv:1909.06639*.

Wei, Z., Wang, X. J., and Wang, D. H. (2012). From distributed resources to limited slots in multiple-item working memory: A spiking network model with normalization. *Journal of Neuroscience*, 32.

Zhang, W. and Luck, S. J. (2008). Discrete fixed-resolution representations in visual working memory. *Nature*, 453.

## 2.8 Appendix

### 2.8.1 Self-Attention in Transformers

Transformers are powerful language models which create contextualized representations of the input sequences. They learn to predict the next token one at a time using an "attention" mechanism that scans other tokens in the sequence for relevant information (Bahdanau et al., 2014). A common practice (that is used here) is to mask any future tokens and so predictions and representations must be made by the current token and any previously seen tokens. These models are able to learn and represent complex sequence modeling tasks.

For a given prediction, Transformer attention generates three separate vectors for each token in the sequence: a query, key, and value ($q$, $k$, $v$). The key vector is a set of tokens that the model has learned are most relevant to the token at hand. The query vector scans the tokens in the key vectors and calculates how much the current prediction should "attend" to each of those tokens. Then, the value vectors at those positions are multiplied by the corresponding weights, summed up, and added to the next representation: for token $i$ at layer $j$, the contextual representation is $\sum_k q_i^j \cdot k_k^j * v_k^j$. Thus, the next token prediction includes earlier sequential information by combining the value vectors from previous tokens. In other words, Transformer attention can be viewed as a read/write mechanism: for a given token, the queries and keys dictate which tokens to read from, the values are the content that is read (proportional to the attention calculated by the keys and queries), and the summed content is written to a new representation at the given token. As we shall see below, the comparison to role-addressable input and output gating operations is evident. The key vectors form addresses analogous to the PFC "stripes" (neural populations that support variable binding in memory). The learned key construction determines the address to store an item and is thus analogous to input gating. Conversely, the query vectors determine which addresses are accessed, and are

thus analogous to output gating.

### 2.8.2 Textual Reference-Back-2 Task

The *textual reference-back task* requires making same/different judgments between incoming symbols assigned to a particular "register" in memory, with respect to those seen previously and linked to those same registers. Like the original tasks, the textual reference-back task is sequential, and requires independent updating and maintenance of two memory registers, each containing one of $S$ arbitrary *symbols* at a time. At the beginning of each sequence, each register is initialized individually to one $s \in S$ (the pool of symbols is shared between registers, which was shown to more substantively tax gating mechanisms in (O'Reilly and Frank, 2006a).) Each sequence is composed of $L$ tuples, each containing register address $\text{Reg}_i$, symbol $\text{Sym}_i$, same/different label $\text{Ans}_i$, and update instruction $\text{Ins}_i$. For a tuple $i \in L$, the **answer** $\text{Ans}_i$ is a binary value that is either Same if symbol $\text{Sym}_i$ is currently stored in the register with address $\text{Reg}_i$, or Different otherwise. The **update instruction** $\text{Ins}_i$ also takes one of two values (`ignore` or `store`), evenly distributed. If the instruction is `ignore`, then the model still needs to make the same/different determination with respect to the stored reference, but the new symbol should not update the register content (i.e., the reference remains unperturbed). If it is `store`, then from that point on in the sequence, $\text{Sym}_i$ is stored in the register with address $\text{Reg}_i$ until otherwise updated. An example is shown in Fig. 2.2.

We implement each reference-back task example in our data as a single sequence, and measure models' ability to predict Same versus Different for each $\text{Ans}_i$. Each sequence has 10 same/different answers, and we generate 100,000 train, 1,000 validation, and 1,000 held-out test sequences.

The class balance of `same` to `different` answer labels in the train/test datasets is roughly 1:2, making a "maximum class" heuristic solution 0.66 accuracy, 0.33 precision, and 0.5 recall. We test several other heuristics, the strongest of which is predicting `same`

if another tuple including `Store` and the target register and target symbol exists in the sequence, which scores 0.80 accuracy, 0.82 precision, and 0.85 recall.

### 2.8.3 Models and Training

We train small, attention-only Transformer models from scratch on our task. Our models contain two decoder-only layers, each with two heads, and no multilayer perceptrons or layer normalization, followed by a linear "unembed" layer to project the output of the last decoder into the space of the entire vocabulary at each timestep In practice, only 'same' and 'different' are ever predicted. Our network uses absolute positional embeddings (Vaswani et al., 2017). The vocabulary contains all possible tokens, represented individually with embedding size $E$. Models are trained to predict the next token with the language modelling objective: if the model is predicting $\text{Ans}_c$, it will have access to all ($\text{Ins}_i$, $\text{Reg}_i$, $\text{Sym}_i$, $\text{Ans}_i$) tuples where $i < c$, as well as $\text{Ins}_c$, $\text{Reg}_c$, and $\text{Sym}_c$. However, the models only receive loss at positions where a same/different token must be predicted (this is similar to the reward function applied in frontostriatal gating networks; (O'Reilly and Frank, 2006b; Soni and Frank, 2024)). Furthermore, each layer gets a causal attention mask– when constructing each token representation, it cannot look ahead at tokens further down the sequence.

The models are trained over 60 epochs of the 100k training data points, learning from 6 million examples in total. Models are evaluated on their accuracy (whether the correct $\text{Ans}_i$ is predicted for each tuple $i$), measured in precision and recall, as well as the `same` versus `different` token logit difference.

### 2.8.4 Path Patching

Path-patching involves making a incisive edit to the representations of a trained model and observing how the model's behavior is affected, allowing one to infer the computations implemented within individual attentional heads (see Fig. 2.7). Path-patching requires a

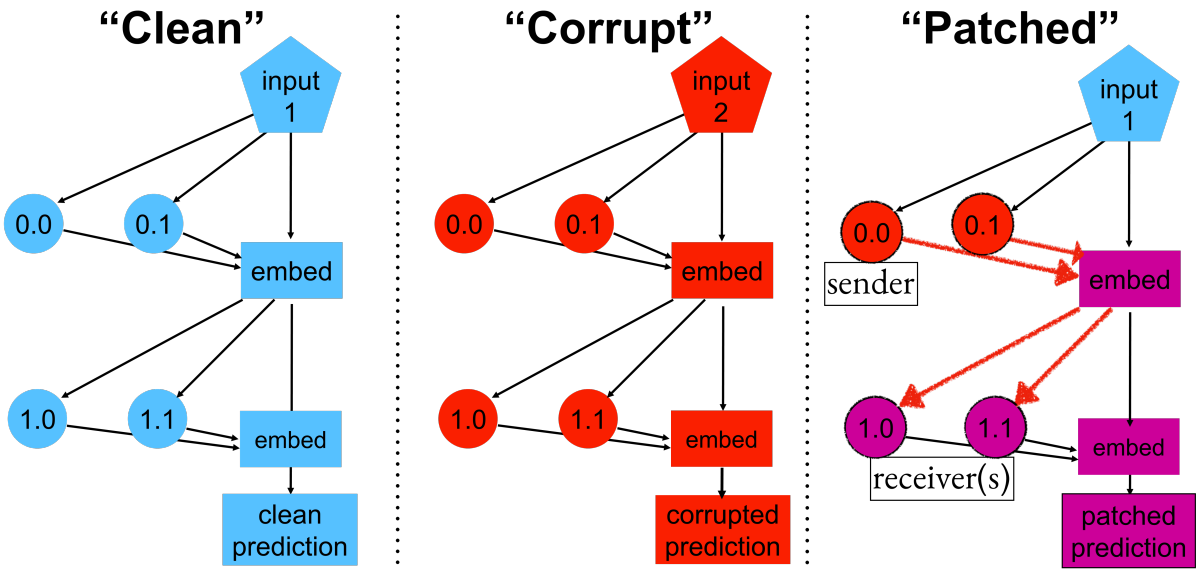



Figure 2.7: **Graphical Path Patching** Graphical diagram of the path-patching process. Attention heads are represented as circles (layer,head index), and contextual representations of each token (as well as the next token prediction) are represented as rectangles.

minimal pair of examples: the "clean" example and the "corrupted" example, in which one token from the clean example is changed, as well as the correct label. Given representations from the model for both the clean and the corrupted examples (the blue and orange components in the figure), we can chose a specific component anywhere in the model (referred to as the "sender"), and replace the clean representation with the corrupted one at that specific component. These embeddings will be received by the next layer ("receiver"), thereby "patching" the path. From there, the patched model will compute the new prediction.

In the figure, we send from layer 0, head 0 and 1 to layer 1, both heads 0 and 1. All clean representations that are not along this path are not modified and are unaffected by the patch. The model then recomputes all representations after the receiver (the "patched" representations), and arrives at a new prediction. If the model output matches the corrupt prediction rather than the clean one, that prediction is causally dependent on the path from sender to receiver. See (Wang et al., 2022) and (Goldowsky-Dill et al., 2023) for a more comprehensive review of path-patching methods.

We perform a small hyperparameter search and select a model that reaches 100% accuracy on the held-out test data for further analysis. We determine the circuit that the model uses through an array of path-patching experiments with a simple minimal pairs paradigm. Our "sender" within path-patching is always both attention heads at layer 0, and our "receiver" is always both attention heads at layer 1.

We first establish that a Transformer model is able to succeed on the reference-back-2 task.

At layer 0, the model learns to condense the task-critical information from each tuple into one embedding, at the position for $\text{Sym}_i$[4]. At this layer, the model pays 85.8% of total attention to the task-critical information to that tuple, and just 14.2% of attention to other tuples.

At layer 1, the attention heads learn to attend to the $\text{Sym}_i$ key vector representing the tuple where information was last stored in the target register. The heads pay 70.2% of total attention to this tuple (the "stored" tuple), and only 29.8% of attention to all other tokens. This behavior is tied to the target register matching the register in the stored tuple, which is analogous to gating of the relevant role-addressable PFC stripe (O'Reilly and Frank, 2006b; Soni and Frank, 2024). We focus our analysis on the Layer 1 representations which exhibit this learned gating policy, shown in Fig. 2.3.

### 2.8.5 Input vs. Output Gating Accuracies, Pretraining

We break down the main analysis into the two types of gating (corresponding to the two subtasks): input and output gating. We see that there is a similar trend across both in regards to accuracy and the model type. Namely, the models pretrained with the 2 register task (mechanistic pretraining), have the highest accuracy. The model with the control task (split set) are less accurate and the worst accuracy are from models

[4]Redundantly, the model does the same at the position for $\text{Ans}_i$. Through additional experimentation, we determine that this is a quirk of Transformer learning, and does not impact our analysis.

that receive no pretraining and are only trained on the main task. These accuracies are calculated at the end of the full training. These results along with other experimental results from human work suggests that output gating seems to play a more important role in this type of working memory tasks.

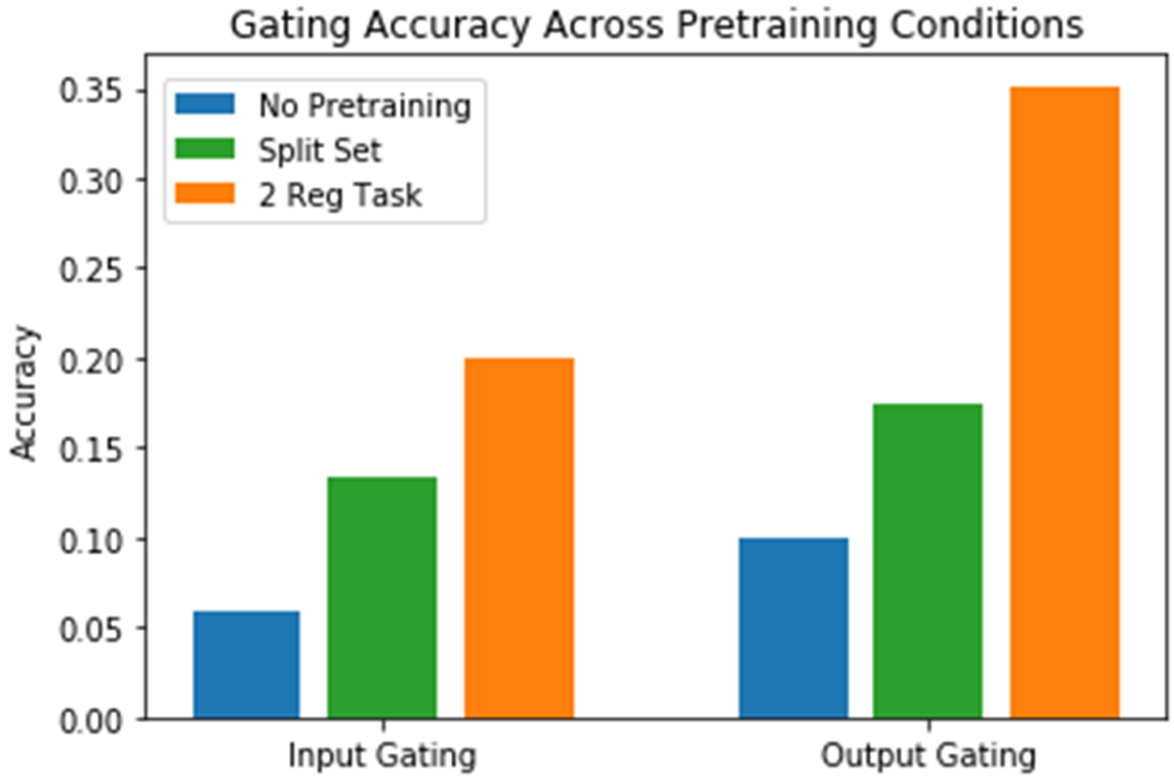



Figure 2.8: **Input and Output Gating Task Accuracy** Pretraining on the 2 register task leads to the highest accuracy on input and output gating subtasks - insinuating that these models learn the most mechanistic solutions. In general the input gating accuracy is lower, indicating a differential role for each gating. There is human experimental evidence to suggest that output gating is harder to learn and is more crucial for better performance

### 2.8.6 Saving Training Time

Pretraining the models with a mechanistic task (red curve in 2.9 allows the model to have fast and high accuracy generalization to the full task (yellow curve). Figure 2.6 shows the models after pretraining, but we can show an advantage of the pretrained models even after accounting for the time it takes to pretrain the models (60 epochs). When comparing a model trained on the full task for 120 epochs to a model that is pretrained

for 60 epochs and trained on the main task for 60 epochs, we see a clear advantage for the pretrained model. When trying to reach a minimum accuracy of 98.5 % (represented by the dotted vertical lines in Figure 2.6), the mechanistic models train faster.

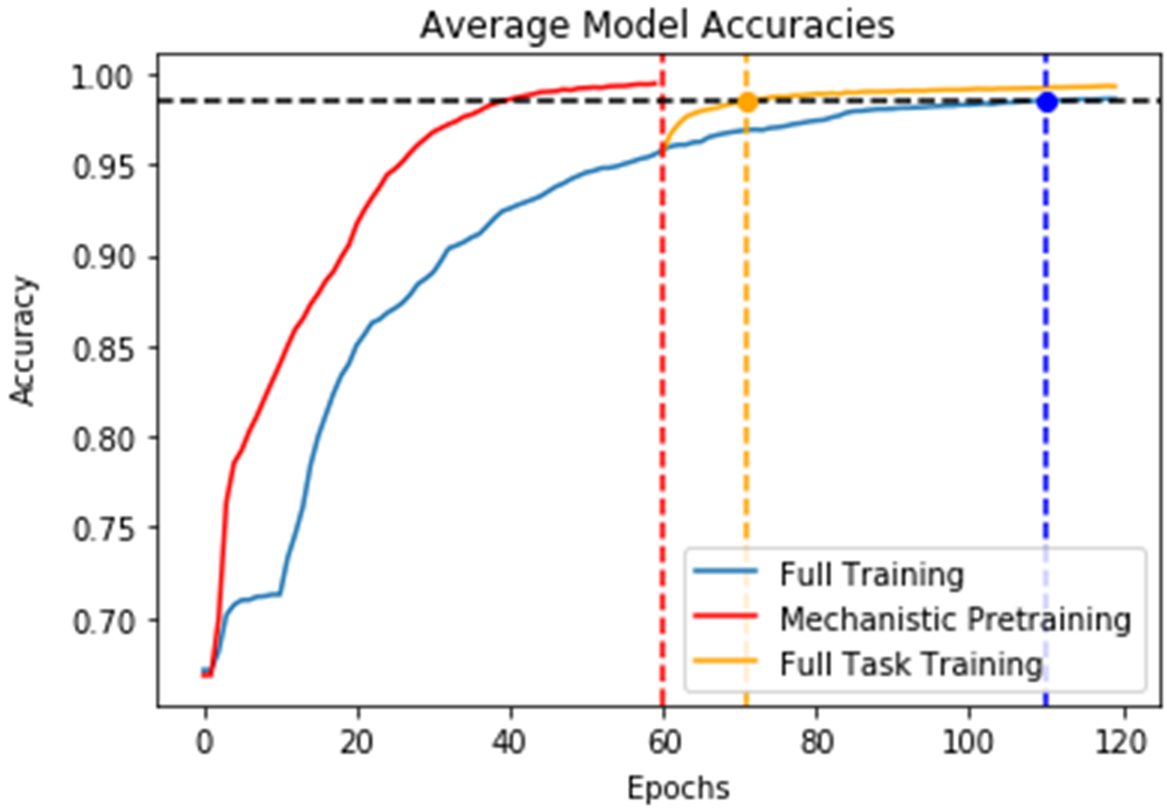



Figure 2.9: **Saved Training Time** The red and yellow curves represent the same models. The red is the mechanistic pretraining and the yellow is the training on the full 3 register task. The blue curve receives no pretraining and is trained on the full 3 register task during the entire duration of the training. The yellow and blue vertical lines depict the first time when the models reach an average accuracy of at least 98.5%, clearly showing the advantage of mechanistic pretraining.

# References


Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Goldowsky-Dill, N., MacLeod, C., Sato, L., and Arora, A. (2023). Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*.

O'Reilly, R. C. and Frank, M. J. (2006a). Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Computation*, 18.

O'Reilly, R. C. and Frank, M. J. (2006b). Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. *Neural computation*, 18(2):283–328.

Soni, A. and Frank, M. J. (2024). Adaptive chunking improves effective working memory capacity in a prefrontal cortex and basal ganglia circuit. *eLife*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, K., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. (2022). Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*.

# CHAPTER 3

# Conclusions and Significance

The literature surrounding WM is vast and has been evolving for quite some time. Subfields of WM include: cognitive components (ex. central executive, phonological loop), neural correlates (ex. PFC, basal ganglia),individual differences (ex. capacity, speed), relationship to other cognitive functions (ex. intelligence, learning), developmental psychology , aging/WM decline, neurological disorders. My work spans multiple subfields and helps our understanding of WM.

My thesis work focuses on working memory in biological and artificial models. My main objective was to better understand WM and what happens to systems (biological and artificial) as task demands change. Analyzing the models' learned gating policies provided valuable information on the models as well as WM in general. I show how learning an adaptive gating strategies is a key, previously undervalued, component to WM. In Chapter 1, I use a biological neural network to combine theoretical models and human experimental work. I show how chunking is a biologically plausible lossy compression method that increases effective capacity. Chunking is supported by the model as WM task demands increase and under a gating policy learned via RL. In Chapter 2, I use an artificial neural network (Transformer) to show how Transformer models also develop stronger gating policies under certain task demands. Learning these gating mechanisms

allow models to generalize faster and better. Lastly, while Transformers do not have a WM capacity limit, when increasing WM load, the model faces learning difficulties similar to biological neural networks. In both cases, successful models learned a proper gating strategy.

### 3.0.1 PBWM: Biological Neural Network Model

In this project I worked to bridge two disparate bodies of work: biological neural networks based in experimental work (PFC, BG, and Thalamus) and theoretical/experimental work on WM capacity (slots, resources, compression - chunking).

Researchers have debated the form of the WM capacity (slots - discrete limit vs. resources - continuous limit) and have conducted many experiments to probe which theory is correct. Experimental work provides some support for both theories, but doesn't resolve the debate. More recently, others have asked how information is encoded (Nassar et al., 2018). Nassar et al. showed that theoretical models that adaptively chunk (i.e. compress information as set size increases) match human experimental data. They also found that humans will chunk more or less based on rewards given on previous trials (RL based). This model can be thought of as a hybrid between slots and resources which can account for data that supports both theories. However, this was an abstract model and some questions remained: was this biological plausible and what would be the underlying mechanisms. In Chapter 1, the project answers those questions: how chunking is implemented using brain-like mechanisms and in a biological neural network. I took a slots-like model and augmented it with the ability to chunk. I designed a chunking mechanism that has characteristic features based in human data. Depending on how much information can be chunked, errors will look more resource-like (items are similar and are chunked, lower precision on some items) or more slots-like (items cannot be chunked, high precision on some items, guessing for other items). This slots-like model showed resource like errors under certain conditions. Under this chunking framework, we can reconcile the slots and

resources theory and provide chunking as a continuum between slots and resource limits.

The biological model we used is the PBWM model. The PBWM (**P**refrontal cortex, **B**asal ganglia **W**orking **M**emory) model (O'Reilly, 1996; O'Reilly and Frank, 2006; Moustafa et al., 2008) learns a gating policy that governs 1) if an item will be gated into memory, 2) where it will be stored, and 3) at time of recall, which information will be output gated. This complete gating policy is learned through training and reinforcement learning, where prior gating actions will be reinforced positively or negatively based on reward on a recall trial. I show how as task demands increase, the model learns a gating policy where the chunked (compressed) representation is preferentially gated into working memory.

We also show that chunking increases **effective capacity** of the network: it can hold information about more items than the allocated capacity. We propose this new terminology **effective capacity** rather than the generally used term of **capacity** because effective capacity encapsulates the fact that there are multiple steps to successful WM recall (input gating, maintenance, and output gating) and we are measuring only what is output by the model/human subject. We are not privy to the "actual" capacity. The effective capacity also fluctuates with task demands (as I showed in Chapter 2). We predict it would also fluctuate with other factors such as training on a certain task or expertise level.

This gating policy is learned using RL, specifically the dopamine signal, which indicates reward prediction error (RPE). Dopamine gain values modulate how much the model will modify gating strategies given a positive or negative reward. I show how learning a proper gating policy relies on a balance of dopamine. Importantly, as task demands increase, with abnormal dopamine balance, the model cannot support chunking and therefore cannot increase its effective capacity to meet task demands. Patient populations (such as Parkinson's and schizophrenic patients) have abnormal dopamine levels as well as notable cognitive deficits that seem to be linked to WM deficits. Here we make a novel prediction

that these patients deficits may not stem just from a neural "capacity" limitation, but rather an inability to manage WM resources.

A simple solution to the WM capacity limit could be to increase the allocated capacity. However after further analysis, I show that increasing capacity $\neq$ better performance. Capacity is not the only determinant of *effective capacity*. When increasing the model's allocated WM capacity (number of stripes), the model has a more difficult credit assignment problem to overcome. It has more possibilities for input and output gating, which requires more training to be able to understand. I show how a model with fewer stripes with chunking ability is better than a model without chunking capabilities and more stripes.

In conclusion, Chapter 1 showed how chunking along with learning an effective gating policy is biologically plausible and an efficient solution that the brain could have adopted.

### 3.0.2 Transformers: Artificial Neural Network Model

Transformers are popular artificial neural networks that have proved to be good at cognitive branching tasks (which in humans typically require WM), without an explicit WM component. Transformers are good at attending to different parts of the input based on relationships between the tokens and using that information to make various inferences, a skill that older neural networks lacked. In humans, this skill is thought to involve sequential attention via WM. In this project, we explore some of the underlying mechanisms learned by the transformer and equate them to input and output gating seen in frontostriatal (PFC-BG) gating.

Transformers (the base of many large language models) have surpassed many neural networks. They do not have recursion or any WM components. Rather, they utilize attention heads to build complex matrices that relate tokens (parts of the input) to one another. The attention head is composed of the key, query, and value matrices. These ideas lend themselves well to the idea of gating. As was shown in Chapter 3, the key construction is similar generating the"addresses" (similar to stripes within the PBWM)

model and input gating. The query vector is similar to output gating.

We use a mechanistic-interpretability method called path-patching. We patch in "corrupted" representations and watch if the model can generate a meaningful response, given the corrupted information. We created two different subtasks (one to probe input gating and one to probe output gating). The WM task used was inspired by the reference back task - a task created to test the need for independent (input and output) gating of multiple items in both humans in PFC-BG models. Most models succeed to learn the task. Remarkably, the models that reach 100% accuracy on the task, all seem to develop input and output gating mechanistic strategies. Further, the learning curves of these models show "grokking": a sharp decrease in training loss, indicative of a model switching from a brittle memorized to a robust mechanistic solution. Other models that perform well, but below 100%, do not exhibit strong gating policies, suggesting they learn more brittle, memorized solutions. The learning curves seem to support this: there is no grokking evidence in these learning curves.

We further show that gating related strategies are related to the task demands. Using two simplified tasks, we show that the gating strategy reflects the task demands. This is not simply an emergent property of Transformers and the multiplicative attention. This task dependent learning may have implications on development and WM training (Buschkuehl et al., 2017).

We hypothesize that learning a more mechanistic solution would allow models to generalize better. We held out symbols from one of the registers and tested performance on a challenge dataset. The challenge dataset included in-distribution and out-of-distribution register-symbol pairings. We found that in the conditions where the model could learn robust input and output gating strategies, the models generalized to out-of-distribution pairings with high accuracy (99.7%). Importantly, in the task setting where no gating is required, the task is learned quickly and with high accuracy. However, in the challenge dataset, these models struggle considerably and due to the brittle solution, even the

performance on in-distribution pairings suffer.

Lastly, we increase the task demands by 1 register and while the task is learned, the performance drops. Given that the Transformer model is intended for language tasks with large sentences, a drop in performance for 1 additional register is concerning. We predict that this drop occurs because, even without a direct WM limit, the model must overcome the credit assignment problem and learn how all the information is related to one another. We believe that pretraining - shaping - the model will ease the model's learning process. Specifically, we predict that pretraining on a task that requires gating to be learned will be most beneficial. We showed both. Namely, pretraining the model was better than no pretraining. Using a pretraining task that required the model to start developing gating strategies allowed for very fast generalization and high accuracy on the new task compared to pretraining on a task that did not require gating. This shows a potential strategy for ML scientists training on difficult tasks: pretrain on a smaller scale task that can push the model to a mechanistic solution.

In conclusion, we show that Transformers exhibit frontostriatal-like input and output gating. We further show that computational limitations in human WM and biological models also hold for Transformers. Connecting the Transformer models with biological models opens up further investigation and better understanding for both of these models.

### 3.0.3 Limitations and Future Work

There are some limitations I would like to address and propose some future work that could address those limitations.

For Chapter 1 and the biological neural network, I am only using 1 chunking stripe for a proof of concept analysis. The principle of using RL to learn the gating strategy (which stripes to use) can be applied to a larger model with multiple chunking stripes (with various degrees of chunking). The connectivity strengths from the input and pfc stripes feeding into the chunk stripe are fixed. However, a more continuous model would include

multiple chunk stripes that could potentially chunk more or less. A meta agent (possibly trained using RL) would select the amount of chunking based on task demands. One could also further explore the architectural options described in 13. In addition to these alternative architectures, one could expand the model to include a visual component.

The biological model has many parameters that were fixed due to various biological constraints. I have explored parameter searching on some of these parameters (e.g. dopamine manipulations), but there is a trove of information one could gather from further analyzing and changing other parameters. This could open up other avenues of exploration such as in other patient populations, understanding WM in aging, and even individual differences in WM. Further, one could better understand the robustness of this model.

In the transformer network, there are also some limitations and possibilities for future work. Transformers solve problems in a non-biological way (ex. input is simultaneously presented, there is no recurrence). We present an analogy between the BG circuit and the input/output gating seen in the Transformers as a method for better understanding underlying computational principles. This analogy helps us differentiate Transformers from other neural networks and could provide insights to improve computational biological models. There is a lot of work left to understand the alternate mechanisms that the models learn. What are the brittle solutions and are there other solutions that resemble gating in different ways? Our path-patching methods allows us to patch in the corrupted embeddings from the first layer. It is possible that the gating is learned in the second layer and we do not see these strategies because we are patching at the first layer. It is not possible to patch at the second layer because we only have a 2-layer model and there is only the unembedding step left. One could also delve further by doing a more thorough hyperparameter search across the transformer networks to understand how robust the gating strategies are. Further, one could develop a grokking phase analysis (Liu et al., 2022). One could also explore other types of generalizations (zero shot learning) and see

how gating mechanisms could help.

### 3.0.4 Conclusions

In my thesis work I hoped to have connected various different theoretical models and experimental data. I also hoped to shift the focus from thinking about WM capacity to thinking about learning how to allocate and utilize WM resources.

# References

Buschkuehl, M., Jaeggi, S. M., Mueller, S. T., Shah, P., and Jonides, J. (2017). Training change detection leads to substantial task-specific improvement. *Journal of Cognitive Enhancement*, 1.

Liu, Z., Kitouni, O., Nolte, N., Michaud, E. J., Tegmark, M., and Williams, M. (2022). Towards understanding grokking: Aneffective theory of representation learning. *NeurIPs*.

Moustafa, A. A., Sherman, S. J., and Frank, M. J. (2008). A dopaminergic basis for working memory, learning and attentional shifting in parkinsonism. *Neuropsychologia*, 46.

Nassar, M. R., Helmers, J. C., and Frank, M. J. (2018). Chunking as a rational strategy for lossy data compression in visual working memory. *Psychological Review*, 125.

O'Reilly, R. C. (1996). Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural Computation*, 8.

O'Reilly, R. C. and Frank, M. J. (2006). Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Computation*, 18.