Working memory in random recurrent neural networks

Anthony Strock

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR
DE L'UNIVERSITÉ DE BORDEAUX

ECOLE DOCTORALE MATHÉMATHIQUES ET INFORMATIQUE

INFORMATIQUE

Par **ANTHONY STROCK**

Mémoire de travail dans les réseaux de neurones récurrents aléatoires

Sous la direction de : **Nicolas ROUGIER**
Co-encadrant : **Xavier HINAUT**

Soutenue le 19 Novembre 2020

Membres du jury :

M. Omri BARAK	Associate Pr.	TECHNION	Rapporteur
M. Peter F. DOMINEY	D.R.	CNRS	Rapporteur
Mme Hélène SAUZÉON	Pr.	Université de Bordeaux	Présidente
Mme Petia KOPRINKOVA-HRISTOVA	Pr.	Bulgarian Academy of Sciences	Examinatrice
M. Xavier HINAUT	C.R.	Inria	Co-encadrant
M. Nicolas ROUGIER	C.R.	Inria	Directeur

Résumé

La mémoire de travail peut être définie comme la capacité à stocker temporairement et à manipuler des informations de toute nature. Par exemple, imaginez que l'on vous demande d'additionner mentalement une série de nombres. Afin de réaliser cette tâche, vous devez garder une trace de la somme partielle qui doit être mise à jour à chaque fois qu'un nouveau nombre est donné. La mémoire de travail est précisément ce qui permettrait de maintenir (i.e. stocker temporairement) la somme partielle et de la mettre à jour (i.e. manipuler). Dans cette thèse, nous proposons d'explorer les implémentations neuronales de cette mémoire de travail en utilisant un nombre restreint d'hypothèses. Pour ce faire, nous nous plaçons dans le contexte général des réseaux de neurones récurrents et nous proposons d'utiliser en particulier le paradigme du *reservoir computing*. Ce type de modèle très simple permet néanmoins de produire des dynamiques dont l'apprentissage peut tirer parti pour résoudre une tâche donnée. Dans ce travail, la tâche à réaliser est une mémoire de travail à porte (*gated working memory*). Le modèle reçoit en entrée un signal qui contrôle la mise à jour de la mémoire. Lorsque la porte est fermée, le modèle doit maintenir son état de mémoire actuel, alors que lorsqu'elle est ouverte, il doit la mettre à jour en fonction d'une entrée. Dans notre approche, cette entrée supplémentaire est présente à tout instant, même lorsqu'il n'y a pas de mise à jour à faire. En d'autres termes, nous exigeons que notre modèle soit un système ouvert, i.e. un système qui est toujours perturbé par ses entrées mais qui doit néanmoins apprendre à conserver une mémoire stable.

Dans la première partie de ce travail, nous présentons l'architecture du modèle et ses propriétés, puis nous montrons sa robustesse au travers d'une étude de sensibilité aux paramètres. Celle-ci montre que le modèle est extrêmement robuste pour une large gamme de paramètres. Peu ou prou, toute population aléatoire de neurones peut être utilisée pour effectuer le *gating*. Par ailleurs, après apprentissage, nous mettons en évidence une propriété intéressante du modèle, à savoir qu'une information peut être maintenue de manière entièrement distribuée, i.e. sans être corrélée à aucun des neurones mais seulement à la dynamique du groupe. Plus précisément, la mémoire de travail n'est pas corrélée avec l'activité soutenue des neurones ce qui a pourtant longtemps été observé dans la littérature et remis en cause récemment de façon expérimentale. Ce modèle vient confirmer ces résultats au niveau théorique. Dans la deuxième partie de ce travail, nous montrons comment ces modèles obtenus par apprentissage peuvent être étendus afin de manipuler l'information qui se trouve dans l'espace latent. Nous proposons pour cela de considérer les *conceptors* qui peuvent être conceptualisé comme un jeu de poids synaptiques venant contraindre la dynamique du réservoir et la diriger vers des sous-espaces particuliers; par exemple des sous-espaces correspondants au maintien d'une valeur particulière. Plus généralement, nous montrons que ces *conceptors* peuvent non seulement maintenir des informations, ils peuvent aussi maintenir des fonctions. Dans le cas du calcul mental évoqué précédemment, ces *conceptors* permettent alors de se rappeler et d'appliquer l'opération à effectuer sur les différentes entrées données au

système. Ces *conceptors* permettent donc d'instancier une mémoire de type procédural en complément de la mémoire de travail de type déclaratif. Nous concluons ce travail en remettant en perspective ce modèle théorique vis à vis de la biologie et des neurosciences.

Introduction Générale

La notion de nombre est l'un des premiers concepts mathématiques abstraits que les enfants apprennent à l'école. En fait, selon l'auteur de l'article, avant même d'aller à l'école, les enfants ont déjà une notion approximative du nombre (Dehaene, 1997). La même notion de nombre, avec les mêmes défauts, que l'on retrouve chez d'autres espèces comme les rats ou les singes. Plus le nombre est grand, moins sa représentation est précise. Cependant, il a été constaté que cette représentation des nombres devient de plus en plus précise au cours de la scolarité. Lorsqu'on demande aux enfants de classer les nombres de 1 à 100 sur une ligne à l'âge de 6 ans (c'est-à-dire à la fin de la maternelle), ils les classent de manière logarithmique, tandis qu'à l'âge de 8 ans (c'est-à-dire à la fin de la deuxième année d'école primaire), ils les classent de manière linéaire (Siegler and Booth, 2004). De plus, en apprenant à compter, les enfants obtiennent une représentation vraiment précise des nombres. Chaque nombre devient associé à un symbole (par exemple, les nombres écrits, les nombres parlés, le nombre de doigts levés dans les mains) qui représente uniquement ce nombre. Ensuite, ils apprendront à utiliser ces représentations précises des nombres pour, par exemple, comparer des quantités, les additionner, les soustraire ou les multiplier. Ces apprentissages passent par différentes étapes. Les enfants apprennent d'abord par cur quelques faits numériques. Ensuite, vu la taille des exemples à connaître, au lieu d'apprendre des exemples par cur, ils apprendront des procédures (c'est-à-dire des stratégies pour décomposer la tâche en sous-tâches plus faciles, et séquentialiser la résolution). Par exemple, lorsque les enfants apprennent à compter, ils doivent apprendre la séquence des nombres entiers naturels. Pour ce faire, ils commencent par apprendre les premiers chiffres (par exemple jusqu'à 10 lorsqu'ils ont 4 ans, jusqu'à 100 lorsqu'ils ont 6 ans, ou jusqu'à 1000 lorsqu'ils ont 8 ans). Mais la séquence des nombres naturels est infinie, il n'est donc pas possible de les apprendre tous par cur. Même avec des nombres plus grands, les enfants de 10 ans apprendront plutôt la décomposition des nombres en milliards, millions, milliers, centaines, dizaines, unités, et le lien entre ces quantités (par exemple, un milliard est mille million, un million est un mille millier, un millier est un dix centaines). De même, lorsque les enfants apprennent l'addition, vers 6 ans, ils apprennent d'abord la somme de deux nombres à un chiffre (aussi appelé table d'addition). Ils peuvent commencer par compter sur leurs doigts, mais très vite, ils devront connaître par cur leur table d'addition. Compte tenu de la symétrie, cela représente quelques dizaines de faits arithmétiques (par exemple, $1 + 1 = 2, 1 + 2 = 3$, etc.) qu'ils connaîtront par cur à l'âge de 6 ans. Cependant, pour les nombres à deux chiffres, il y aurait 100 fois plus de faits arithmétiques à connaître, et encore 100 fois plus pour chaque chiffre supplémentaire. Au lieu d'apprendre par cur, ils apprendront de nombreuses techniques. Par exemple, pour ajouter 9 à 27, ils pourraient décomposer 9 en 10 - 1. L'ajout de 10 augmenterait les dixièmes chiffres de 1, la suppression de 1 diminuerait le chiffre de l'unité de 1. Enfin, pour faire des sommes dans le cas le plus général, à l'âge de 8 ans, ils apprendraient un algorithme général. Par exemple, ils ajouteraient d'abord les chiffres de l'unité, puis les dixièmes deux chiffres avec le report potentiel, puis

les centaines de chiffres avec le report potentiel, etc.

Au début, les enfants apprennent à mettre en uvre de telles procédures en utilisant un support externe pour la mémoire (par exemple en utilisant leurs doigts pour compter, en utilisant un stylo et du papier pour décomposer une addition). Mais ce qui est intéressant dans le contexte de ma thèse, c'est que plus tard, ils apprennent également à les faire "dans leur tête", c'est-à-dire sans aucun support externe pour la mémoire. Par exemple, ils arrêteront d'utiliser leurs doigts pour compter, ou ils n'auront plus besoin d'un stylo et d'un papier pour ajouter des nombres à plusieurs chiffres. En fait, leur capacité à retenir temporairement des informations dans leur tête ne cessera de s'améliorer tout au long de leur enfance (Gathercole, 1999). Entre 3 et 16 ans, leur performance dans diverses tâches impliquant la conservation temporaire d'informations fera plus que doubler. Il est intéressant de noter que cette amélioration cognitive coïncide avec le développement d'une de leurs régions cérébrales : le cortex préfrontal (PFC) (Uytun, 2018).

La capacité de conserver et de manipuler temporairement des informations dans leur tête est également connue sous le nom de mémoire de travail (WM). D'une certaine manière, la mémoire de travail peut être considérée comme une extension de la mémoire à court terme. La mémoire à court terme se concentre uniquement sur le stockage temporaire de l'information (Atkinson and Shiffrin, 1968). Cependant, on s'est rendu compte qu'il était difficile de démêler le stockage temporaire de l'information et son traitement. Les deux sont liés et s'influencent mutuellement (Craik and Lockhart, 1972). C'est pourquoi le concept de mémoire de travail a été inventé par Baddeley and Hitch pour combiner à la fois le stockage temporaire et le traitement (Baddeley and Hitch, 1974). Depuis lors, de nombreux chercheurs ont essayé de comprendre son fonctionnement. Certains d'entre eux visent à identifier ses propriétés/limites, et maintenant une pléthore d'effets sont connus. A tel point qu'il est devenu difficile d'identifier quelles sont ses propriétés les plus importantes à modéliser. Récemment, Oberauer et al. a rassemblé ces effets et les données les montrant et a essayé de guider la modélisation en classant leur importance (Oberauer et al., 2018). Afin d'aller plus loin dans la compréhension de la mémoire de travail, un autre axe de recherche est mis en place pour identifier comment elle est mise en uvre dans le cerveau. En fait, il n'y a pas que les humains qui ont une mémoire de travail, beaucoup d'autres espèces en ont une et en particulier les primates non humains. En étudiant le cerveau de ces primates non humains, une région cérébrale a été identifiée comme cruciale pour la mémoire de travail : le cortex préfrontal (PFC) (Funahashi, 2017a). Encore une fois, il s'agit de la même région cérébrale. En effet, il a été noté que des dommages au PFC entraînent des déficits de la mémoire de travail (Funahashi, 2017a). De plus, des enregistrements dans le PFC ont montré des représentations de la mémoire de travail (Funahashi, 2017a). Mais le rôle du PFC n'est pas encore complètement compris et de nombreuses autres régions du cerveau, y compris les régions corticales et sous-corticales, semblent jouer un rôle (Christophel et al., 2017). En outre, la manière dont le PFC maintiendrait ou aiderait à maintenir l'information n'est pas encore totalement comprise non plus. Une hypothèse qui est soutenue par des preuves empiriques est que pour maintenir l'information, les neurones le font par une activité neuronale soutenue (Leavitt, Mendoza-Halliday, and Martinez-Trujillo, 2017). Mais dans certains contextes, il a été constaté que ce n'était pas le cas. Cela suggère qu'il existe des moyens de maintenir l'information d'une manière silencieuse par rapport à l'activité. Une autre hypothèse suggère donc que l'information pourrait être maintenue à court terme de

manière synaptique (Mongillo, Barak, and Tsodyks, 2008a). Une autre hypothèse encore propose que l'information pourrait être encodée de manière plutôt dynamique, c'est-à-dire que la représentation de l'information détenue dans les neurones change au fil du temps (Stokes, 2015a).

La plupart des modèles de calcul de la mémoire de travail proposent de fabriquer des attracteurs avec les neurones (par exemple, des attracteurs plans (Amari, 1977), des attracteurs en forme d'anneau (Compte, 2000), des attracteurs en forme de ligne (Lim and Goldman, 2013a)). Ils supposent donc implicitement que les neurones utilisent une activité soutenue pour maintenir l'information. S'il existe quelques modèles de calcul pour la mémoire de travail synaptique (Mongillo, Barak, and Tsodyks, 2008a), à notre connaissance, aucun modèle de calcul n'a tenté d'expliquer comment l'encodage peut être dynamique, ni même comment ces différentes manières de maintenir (soutenu, dynamique, synaptique) peuvent coexister/interagir. De plus, la plupart des modèles se concentrent sur la manière dont les neurones peuvent maintenir et non sur la manière dont les neurones décident quand et quoi maintenir, c'est-à-dire sur l'aspect procédural de la mémoire de travail. Le seul dont nous ayons connaissance est celui de O'Reilly and Frank qui repose sur un circuit complexe impliquant de nombreuses régions du cerveau, dont le PFC (O'Reilly and Frank, 2006a).

Le premier objectif majeur de cette thèse est de proposer un mécanisme neuronal qui utilise un encodage dynamique pour maintenir l'information et identifier ce qui peut faire passer son comportement de soutenu à dynamique. Contrairement aux modèles actuels, pour trouver le mécanisme neuronal potentiel, nous évitons de décider a priori quel mécanisme est utilisé pour maintenir. Nous profitons de la capacité des réseaux neuronaux récurrents (RNN) à apprendre des tâches. Nous utilisons une classe très simple de RNN, appelés réservoirs, qui semblent avoir des propriétés similaires à celles du PFC (Enel et al., 2016a). Nous enseignons à ces réservoirs comment résoudre une tâche de ″gating", puis nous disséquons le mécanisme que ce réseau utilise pour maintenir l'information. Ce ″gating" pourrait être considéré comme un mécanisme générique de maintien de l'information. Il combine deux sous-mécanismes : (1) un mécanisme de porte qui s'ouvre lorsque de nouvelles informations doivent être stockées et se ferme dans le cas contraire, et (2) un mécanisme qui maintient (resp. met à jour) les informations lorsque la porte est fermée (resp. ouverte). Ces deux mécanismes pourraient être mis en uvre par deux régions cérébrales différentes (O'Reilly and Frank, 2006a). Le ganglion basal (BG) effectuerait le premier tandis que le PFC effectuerait le second. Nous explorons ici le second mécanisme, c'est-à-dire celui qui pourrait être exécuté par le PFC. En fait, nous allons un peu plus loin en essayant de relier une éventuelle mémoire de travail synaptique à la mémoire de travail neurale par activité que nous avons proposée. À cette fin, nous utilisons des concepteurs qui ont été introduits comme modèle pour les mémoires à long terme de modèles temporels (Jaeger, 2017a).

Cette thèse est sur article et est structurée comme suit. Tout d'abord, nous faisons une revue complète de la littérature combinant (1) les études expérimentales de la mémoire de travail en neurosciences et en psychologie (voir Chapitre 2), (2) la description théorique de la mémoire de travail allant des abstraits comme celui de Baddeley aux précis avec des systèmes dynamiques (voir Chapitre 3), et (3) une description historique de l'évolution des réseaux neuronaux récurrents pour surmonter un problème analogue à un problème de mémoire de travail (voir Chapitre 4). L'examen complet est directement suivi de deux chapitres de contribution. Dans

le chapitre 5, nous nous concentrons sur un mécanisme neuronal de "gating" (publié (Strock, Hinaut, and Rougier, 2020)). Dans le chapitre 6, nous explorons une mémoire de travail synaptique (en révision (Strock, Rougier, and Hinaut, 2020)). Enfin, dans le chapitre 7, nous résumons les conclusions tirées de notre approche, nous discutons ses limites ainsi que ses perspective.




THESIS PRESENTED

IN ORDER TO OBTAIN THE TITLE OF

DOCTEUR
DE L'UNIVERSITÉ DE BORDEAUX

ECOLE DOCTORALE MATHÉMATHIQUES ET

INFORMATIQUE

INFORMATIQUE

By **ANTHONY STROCK**

Working Memory in Random Recurrent Neural Networks

Under the direction of : **Nicolas ROUGIER**
Co-supervisor : **Xavier HINAUT**

Defended on November 19, 2020

Jury members :

M. Omri BARAK	Associate Pr.	TECHNION	Reviewer
M. Peter F. DOMINEY	D.R.	CNRS	Reviewer
Mme Hélène SAUZÉON	Pr.	Université de Bordeaux	President
Mme Petia KOPRINKOVA-HRISTOVA	Pr.	Bulgarian Academy of Sciences	Examiner
M. Xavier HINAUT	C.R.	Inria	Co-supervisor
M. Nicolas ROUGIER	C.R.	Inria	Director

Abstract

Working memory can be defined as the ability to temporarily store and manipulate information of any kind. For example, imagine that you are asked to mentally add a series of numbers. In order to accomplish this task, you need to keep track of the partial sum that needs to be updated every time a new number is given. The working memory is precisely what would make it possible to maintain (i.e. temporarily store) the partial sum and to update it (i.e. manipulate). In this thesis, we propose to explore the neuronal implementations of this working memory using a limited number of hypotheses. To do this, we place ourselves in the general context of recurrent neural networks and we propose to use in particular the *reservoir computing* paradigm. This type of very simple model nevertheless makes it possible to produce dynamics that learning can take advantage of to solve a given task. In this job, the task to be performed is a gated working memory task. The model receives as input a signal which controls the update of the memory. When the door is closed, the model should maintain its current memory state, while when open, it should update it based on an input. In our approach, this additional input is present at all times, even when there is no update to do. In other words, we require our model to be an open system, i.e. a system which is always disturbed by its inputs but which must nevertheless learn to keep a stable memory.

In the first part of this work, we present the architecture of the model and its properties, then we show its robustness through a parameter sensitivity study. This shows that the model is extremely robust for a wide range of parameters. More or less, any random population of neurons can be used to perform *gating*. Furthermore, after learning, we highlight an interesting property of the model, namely that information can be maintained in a fully distributed manner, i.e. without being correlated to any of the neurons but only to the dynamics of the group. More precisely, working memory is not correlated with the sustained activity of neurons, which has nevertheless been observed for a long time in the literature and recently questioned experimentally. This model confirms these results at the theoretical level. In the second part of this work, we show how these models obtained by learning can be extended in order to manipulate the information which is in the latent space. We therefore propose to consider *conceptors* which can be conceptualized as a set of synaptic weights which constrain the dynamics of the reservoir and direct it towards particular subspaces; for example subspaces corresponding to the maintenance of a particular value. More generally, we show that these *conceptors* can not only maintain information, they can also maintain functions. In the case of mental arithmetic mentioned previously, these *conceptors* then make it possible to remember and apply the operation to be carried out on the various inputs given to the system. These *conceptors* therefore make it possible to instantiate a procedural working memory in addition to the declarative working memory. We conclude this work by putting this theoretical model into perspective with respect to biology and neurosciences.

Contents

Chapter 1

General Introduction

The notion of number is one of the first abstract mathematical concepts that kids learn in school. Actually, according to Dehaene, even before going to school kids already have an approximate notion of number (Dehaene, 1997). The same notion of number, with the same flaws, that can be found in other species such as rats or monkeys. The larger the number, the less accurate its representation is. However, it has been found that this representation of numbers becomes more and more accurate during schooling. When kids are asked to arrange the numbers from 1 to 100 on a line by the age of 6 (i.e. end of kindergarten), they will arrange them logarithmically, while by the age of 8 (i.e. end of second grade) they will arrange them linearly (Siegler and Booth, 2004). Moreover, as they learn to count, kids obtain a truly precise representation of numbers. Each number becomes associated with a symbol (e.g. written numbers, spoken numbers, number of fingers lifted in the hands) that represents only this number. Afterward, they will learn how to use these accurate representations of numbers to, for example, compare quantities, add, subtract, or multiply them. These learnings go through different stages. First kids will learn by heart a few numerical facts. Then, given the size of the examples to be known, instead of learning examples by heart, they will learn procedures (i.e. strategies to break down the task into easier tasks and sequentialize the resolution). For example, when kids learn to count, they have to learn the sequence of natural numbers. To do so, they first start by learning the first few numbers (e.g. up to 10 when they are 4 years old, up to 100 when they are 6 years old, or up to 1000 when they 8 years old). But the sequence of natural numbers is infinite, it is thus not possible to learn them all by heart. Even with bigger numbers 10-year-old kids will rather learn the decomposition of number in billions, millions, thousands, hundreds, tens, units, and the link between these quantities (e.g. one billion is thousand million, one thousand is ten hundred). Similarly, when kids learn addition, around 6 years old they learn first all the sum of two single-digit numbers (a.k.a. addition table). They may start by counting on their fingers, but very soon they will have to know their addition table by heart. Given the symmetry, that's a few dozen arithmetic facts (e.g. $1+1=2$, $1+2=3$, etc.) that they will know by heart by the age of 6. However for double-digit numbers, it would be 100 times more arithmetic fact to know, and another 100 times more for each additional digit. Instead of learning by heart, they will learn many techniques. For instance, to add 9 to 27, they could decompose 9 into 10 - 1. Adding 10 would increment the tenth digits by 1, removing 1 would decrement the unit digit by 1. In the end, to make sums in the most general case, at the age of 8 they would learn a general algorithm. For instance, they would first add the unit digits, and then add the tenth two-digits with the potential carry over, and then add the hundreds digits with the potential carryover, etc.

In the beginning, kids learn to implement such procedures by using external support for memory (e.g. using their fingers to count, using pen and paper to break down an addition). But what is interesting in the context of my thesis is that later on they also learn to do them "in their heads", i.e. without any external support for memory. For instance, they would stop using their fingers to count, or they wouldn't need anymore a pen and a paper to add multi-digit numbers. In fact, their ability to temporarily hold information in their heads will keep improving throughout their childhood (Gathercole, 1999). Between the ages of 3 and 16, their performance on various tasks involving maintaining temporary information will more than double. Interestingly, this cognitive improvement coincides with the development of one of their brain region: the prefrontal cortex (PFC) (Uytun, 2018).

The ability to temporarily hold and manipulate information in their heads is also known as working memory (WM). In some way, working memory can be seen as an extension of short-term memory. Short-term memory focuses only on the temporary storage of the information (Atkinson and Shiffrin, 1968). However, it was realized that it was difficult to untangle the temporary storage of information, and its processing. The two are interlinked and influence each other (Craik and Lockhart, 1972). Hence the concept of working memory has been coined by Baddeley and Hitch to combine both temporary storage and processing (Baddeley and Hitch, 1974). Since then a lot of researchers have tried to understand its functioning. Some of them is aiming at identifying its properties/limitations, and now a plethora of effects are known. So much so that it became difficult to identify which are its most important properties to model. Recently, Oberauer et al. collected these effects and the data showing them and tried to guide the modeling by ranking their importance (Oberauer et al., 2018). In order to go further in the understanding of working memory another line of research aimed at identifying how it is implemented in the brain. In fact, it is not just humans who have a working memory, many other species have and especially non-human primates. By studying the brain of such non-human primates, a brain region has been identified as crucial for working memory: the prefrontal cortex (PFC) (Funahashi, 2017a). Yet again the same brain region. Indeed, it has been noted that damage to the PFC leads to working memory deficits (Funahashi, 2017a). Moreover, recordings in the PFC have exhibited working memory representations (Funahashi, 2017a). But the role of PFC is not yet completely understood and a lot of other regions of the brain including cortical and subcortical regions seem to play a role (Christophel et al., 2017). Beyond that, how the PFC would maintain or help maintain information is not yet fully understood either. One hypothesis that is supported by empirical evidence is that in order to maintain information, neurons do it through sustained neural activity (Leavitt, Mendoza-Halliday, and Martinez-Trujillo, 2017). But in some contexts, it has been noted that this was not the case. This suggests that there are ways of maintaining information in an activity-silent way. Another hypothesis therefore suggests that the information could be maintained in the short-term in a synaptic way (Mongillo, Barak, and Tsodyks, 2008a). Alternatively, yet another hypothesis proposes that the information could be encoded in a rather dynamic way instead, i.e. the representation of the information held in the neurons changes over time (Stokes, 2015a).

Most computational models of working memory propose how to handcraft attractors with neurons (e.g. plane attractors (Amari, 1977), ring attractors (Compte, 2000), line attractors (Lim and Goldman, 2013a)). They will therefore implicitly assume that neurons use sustained activity to maintain information. If there are some computational models for synaptic working memory (Mongillo, Barak, and Tsodyks,

2008a), to the best of our knowledge, not any computational model have tried to explain how the encoding can be dynamic, or even how these different ways of maintaining (sustained, dynamic, synaptic) can coexist/interact. Moreover, most models focus on how neurons can maintain and not on how neurons decide when and what to maintain, i.e. on the procedural aspect of working memory. The only one we are aware of is the one of O'Reilly and Frank which relies on a complex circuitry involving many brain regions including PFC (O'Reilly and Frank, 2006a).

The first major aim of this thesis is to propose a neural mechanism that uses a dynamic encoding to maintain information and to identify what can switch its behavior from sustained to dynamic. Unlike current models, to find the potential neural mechanism we avoid handcrafting. We take advantage of the ability of Recurrent Neural Networks (RNN) to learn tasks. We use a very simple class of RNN, a.k.a. reservoirs, which appear to have properties similar to those of PFC (Enel et al., 2016a). We teach these reservoirs how to solve a gating task, and then we dissect what mechanism this network takes to maintain information. This gating could be considered as a generic mechanism for maintaining information. It combines two sub-mechanisms: (1) a mechanism that acts as a gate, opens when new information needs to be stored and closes otherwise, and (2) a mechanism that maintains (resp. updates) information when the door is closed (resp. open). It has been hypothesized that the two mechanisms could be carried out by two different brain regions (O'Reilly and Frank, 2006a). The Basal Ganglia (BG) would perform the first while the PFC would perform the second. Here we explore the second mechanism, i.e. the one that could be performed by the PFC. In fact, we are going a little further by making an attempt to link a possible synaptic based working memory with the neural activity-based working memory we proposed. For this purpose, we use Conceptors that has been introduced as a model for long-term memories of temporal patterns (Jaeger, 2017a).

This thesis is article-based, and it is structured as follows. First, we make a comprehensive review of the literature combining (1) the experimental studies of working memory in neuroscience and in psychology (see Chapter 2), (2) the theoretical description of working memory ranging from abstract ones like Baddeley's to precise ones with dynamical systems (see Chapter 3), and (3) a historical description of how recurrent neural networks evolved to overcome a problem analog to a working memory problem (see Chapter 4). The comprehensive review is directly followed by two contribution Chapters. In Chapter 5 we focus on a neural mechanism for gating (published (Strock, Hinaut, and Rougier, 2020)). In Chapter 6 we explore possible synaptic mechanism for working memory (in revision (Strock, Rougier, and Hinaut, 2020)). Finally in Chapter 7 we summarize the conclusions drawn from our approach, we discuss its limitation as well as its perspective.

Chapter 2

Working memory: Empirical

The primary purpose of this chapter is to clarify what is working memory (see Section 2.1). The second is to give a global view of how working memory is studied experimentally (see Section 2.2), and a global view of what are the most important experimental findings concerning working memory, both behaviourally (see Section 2.3) and neurophysiologically (see Section 2.4).

2.1 The definitions of working memory

Working memory is a concept that has evolved over the years (Cowan, 2008; Cowan, 2016). Already at the end of the 80s James proposed that, like computer memory, human memory can be categorized in two types (James, 1890): (1) the primary memory, a small volatile memory representing the conscious present, and (2) the secondary memory, a big and less volatile memory representing the knowledge stored over a lifetime. Later, Atkinson and Shiffrin revisited this distinction by clarifying it in terms that are less related to computers (Atkinson and Shiffrin, 1968). Atkinson and Shiffrin proposed the concept of short-term memory as a memory where information is stored in a very accessible state temporarily. If primary memory and short-term memory are similar concepts they differ in their implication of consciousness/awareness. For Atkinson and Shiffrin, consciousness/awareness has no main role in short-term memory. It is possible that information can be accessed without being or having been part of the conscious present. Without distinction, strictly speaking, with short-term memory, the term working memory has been coined in the literature by Miller, Galanter, and Pribram (Miller, Galanter, and Pribram, 1960). It was then popularized later by Baddeley and Hitch with a slightly different meaning (Baddeley and Hitch, 1974). For Baddeley and Hitch, short-term storage of information and its processing should not be dissociated. They are two sides of the same coin, what they call working memory.

To ground working memory to something concrete, let us illustrate its role with an example. When a person is asked to verbally solve a mental calculation such as 27+9, many different strategies can be used. For example, she could decompose 9 into $10 - 1$, and thus increase the tenth digit and decrease the unit digit. She could also decompose 9 in $3 + 6$ so that she took enough from 9 to fill 27 in 30, and the remaining would be the unit. Alternatively, she could recognize 27 as a multiple of 9 (i.e. $27 = 3 \times 9$), and interpret the question as "what is the next multiple of 9 after 27?". However, whichever strategy, before doing the math, the person must remember the calculation she was asked to make. Concretely, this means that this

person has to remember the audio stimulus that she experienced in the absence of this stimulus. Then, to do the math, the person will choose a strategy. For the sake of the example, let us say that she chooses the classical addition algorithm she learned in school. Then, she would start by adding 7 and 9, remember 6 and the carry-over, then add 2 and the carry-over, remember 3, and finally answer 36. During this procedure, which takes place inside her head, we can see that the information she maintains is evolving. Before arriving at the solution, she has to go through partial results that she has to remember at the end in order to respond. She must also remember intermediate information, such as the carry-over, which becomes useless and discarded long before the final result is found. After some time, e.g. several days, if one tries to ask that person again for the result of the calculation that she was asked, she will probably not be able to answer. This person may not even remember what calculation she was asked to do, or even less every step she went through to solve that mental calculation. To sum up, (1) to do the mental calculation the person remembered information, and (2) when it was no longer useful she forgot it. This shows that there is information that people only maintain for a short time, e.g. when it is useful. That is not only true for people, most animals are also capable of holding temporary information. The information that is maintained can be as well related to perception (e.g. visual or auditory stimuli) as abstract (e.g. intermediate calculations required to solve some mental arithmetic).

Since the term working memory has been coined by Miller, Galanter, and Pribram it has been used with various different definitions (Cowan, 2016). Recently, Cowan gathered and compared nine of them (Cowan, 2016). He has highlighted five main features that distinguish them: (1) does working memory include passive storage? , (2) does working memory include processing?, (3) are long-term memory interaction part of working memory?, (4) does working memory have limited capacity?, and (5) is working memory multi-component in nature? Let us use the former mental calculation example to illustrate the first three. (1) In the mental calculation example, all the temporary memories used to do the math are tightly linked to consciousness/awareness, i.e. when the memory is retained, the person is aware that the memory is retained. However, by using subliminal visual stimuli, i.e. that last for a very short time ($\approx$ 20ms), Soto, Mäntylä, and Silvanto have shown that information can be stored temporarily and used without consciously doing so, hence without any kind of rehearsal or reactivation (Soto, Mäntylä, and Silvanto, 2011). The question is therefore whether the same mechanisms are used in both cases? Should a distinction be made? For a recent review on conscious/unconscious working memory see (Velichkovsky, 2017). (2, 3) In the mental calculation example, doing the calculation does not only involve temporary storage but also the processing of the information that is stored (e.g. the sum algorithm used) or access to long-term memory (e.g. addition tables learned at school). Initially, when Miller, Galanter, and Pribram used the term working memory, there was no clear difference between short-term storage and working memory (Miller, Galanter, and Pribram, 1960). The distinction was rather between long-term and short-term memories because of their different properties, e.g. temporal decay (Cowan, 2008). However, over the years, the term working memory has evolved to also describe the whole system that temporarily maintains information and processes that information.

This variety of definitions of working memory results in a wide range of experimental studies. However, that is not the only reason why working memory experimental literature is so vast. For instance, as working memory can hold information of different nature (e.g. auditory, visual, olfactory, tactile, ...), each different nature requires

its own experimental paradigm. Moreover, the objectives of working memory experimental studies vary a lot. Some studies aim at characterizing the limits of human working memory, and thus they will use more and more complex tasks, whereas some others use animal models (e.g. monkeys, rats) to discover the neural foundation of working memory, and they will focus first on simpler tasks. The goal of the following section is to give an overview of the principal experimental paradigms, i.e. tasks performed and data collected, and why they are used. To build this overview the following three main recent reviews were used: (Oberauer et al., 2018) closer to psychology, (Funahashi, 2017a) closer to neuroscience, (Chai, Hamid, and Abdullah, 2018) at the interface between psychology and neuroscience.

2.2 Experimental paradigms

Working memory tasks are designed such that the expected response depends on what should be stored in/processed by working memory. Generally speaking, we can distinguish between three different types of working memory experimental paradigm: (1) temporary storage of a single stimulus, (2) temporary storage of a sequence of stimuli, and (3) temporary storage of several stimuli and its internal processing. As the name suggests, the first two types focus only on the temporary storage of information whereas in the third type the emphasis is not only on the temporary storage of information but also on the inner manipulation of the content that is stored.

In order to have a period where only one stimulus is maintained, an explicit delay is added between the arrival of the first stimuli and the expected response. Indeed, since there is a delay between the first stimulus and the response and because the response depends on this stimulus, it is expected that something about the stimulus will be held in memory during this delay. Moreover, during this delay, no stimuli are given, i.e. the first stimulus is not given anymore and no other stimuli are provided. There is thus, in theory, a moment when the first stimulus and only the first stimulus is held in memory. That is why this paradigm is the most widely used in animal studies. To understand how the brain holds temporary information, one can start by understanding how within this delay the brain holds a stimulus. This paradigm is common to several tasks such as for example the delayed match to sample task (DMTS, see Figure 2.1) (Aggleton, Hunt, and Rawlins, 1986). In this task, a stimulus is presented briefly, followed by a delay of several seconds to a few minutes where no additional stimuli are received. After this delay, the same stimulus and a different one are presented simultaneously and the animal must choose the one presented before the delay to get a reward. In order to recognize which of the two is the correct stimuli, the animal should somehow have held in memory something which helps to recognize the first stimulus. There are multiple variants of these delay tasks, and some of them are even studied in humans. For instance, in the change detection task (CD) (Wilken and Ma, 2004) the subject receives a first stimulus, a delay is introduced and then the subject receives a second stimulus and must determine whether or not the second stimulus is the same than the first one. The main difference is that the two stimuli are not presented simultaneously after the delay. However, the overall idea is very similar.

Many types of information can be stored in memory during the delay but in the present work, we focused on information evolving in a continuous space such as for example in the delayed vibrotactile comparison task (see Figure 2.2) (Barak et al.,

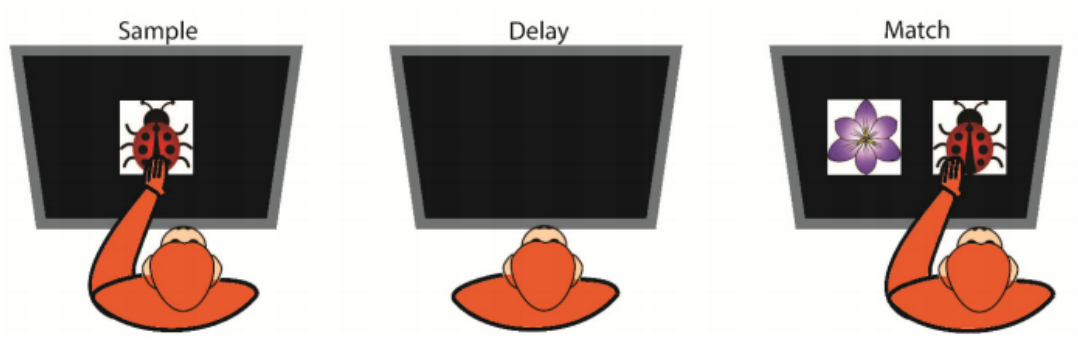


FIGURE 2.1: Delayed match to sample task. (Left) Sample: First, the monkey is shown a visual stimulus on a screen, here a ladybug picture. (Middle) Delay: Then, the monkey waits in front of an empty screen for several seconds or even a few minutes. (Right) Match: Finally, two visual stimuli are shown, and the monkey has to recognize and select the stimuli he has seen before the delay. Here, on the left, the monkey is shown a flower picture, and on the right a ladybug picture, and he has to select the ladybug picture. Figure extracted from (Curry et al., 2017).

2013). In this task, a vibrational stimulus is applied to the hand of a monkey followed by a delay. After this delay, another vibrational stimulus is applied and the monkey has to decide which of the first or second stimuli was vibrating at the highest frequency. In order to compare both stimuli, the monkey has to remember the frequency of the first stimulus during the delay period. In theory, these frequencies could possibly evolve continuously from 0 to infinity. That is precisely why the information the monkey has to store evolve in a continuous space. Similar vibrotactile tasks have also been studied in humans (Bancroft, Hockley, and Servos, 2011). In humans, color has been used as a continuous dimension. For instance, in the continuous reproduction task (CR, see Figure 2.3), a.k.a. delayed estimation, the color is associated with a position on a ring. Many colored circles are shown at different positions followed by a delay. After the delay, the position of one of the circles is highlighted and the subject has to reproduce its color on a ring representing all the colors.

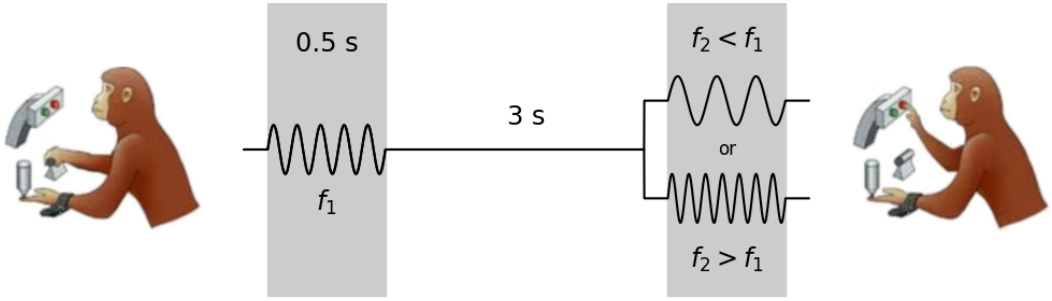


FIGURE 2.2: Delayed vibrotactile comparison task. A vibrational device is put in the hand of a monkey. This device will vibrate at a given frequency $f1$ for 0.5 second, then it will stop vibrating for a delay of 3 seconds. After the delay, it will start again vibrating again but now at a frequency $f2$ for 0.5 second. In the end, the monkey has to determine whether the device was vibrating faster the first time ($f1 > f2$) or the second time ($f2 > f1$). Figure adapted from (Romo and Salinas, 2003).

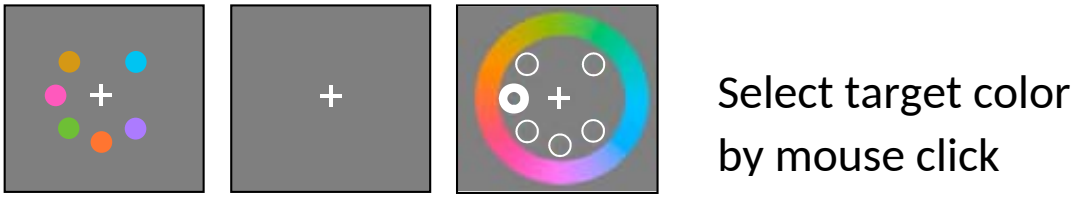


FIGURE 2.3: Continuous reproduction task. (Left) Several colored circles are displayed on a screen. (Middle) The circles are removed from the screen and the screen is left empty during a delay period. (Right) The location of one of the circle is emphasized and the participant has to select its color on a color wheel. Figure extracted from (Oberauer et al., 2018).

In the previous tasks, in the delay, the animal/human does not receive any stimulus. There are, however, tasks where stimuli keep coming during the delay. In such tasks, several stimuli are given one after the other, and the subject must maintain the sequence of stimuli or part of it. There is thus an effective delay between the first stimuli and the response, but no time interval without a stimulus being received. After the presentation of the sequence of stimuli, the way the participant is asked to recall the sequence might vary from task to task. In the serial recall task (SR), the subject must recall the sequence of stimulus in order of presentation (forward or backward) (Unsworth and Engle, 2006) while in the free recall task (FR), the subject must recall the sequence of stimulus in any order (Rohrer and Wixted, 1994). In the probed recall (PR), a retrieval cue uniquely identifying one of the stimuli (e.g. an ordinal position) is given and the subject must recall the stimulus associated with it (Murdock, 1968). The recall can also be requested each time a new stimulus is given. For instance, in the n-back task (NB), each time a new stimulus is given, the subject must identify whether or not the stimulus matches the one received n steps back (Jonides et al., 1997).

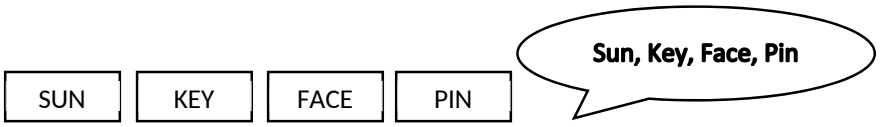


FIGURE 2.4: Serial recall task. A list of items are presented one after the other. After they have been all presented the participant is asked to recollect them in order of presentation. Figure extracted from (Oberauer et al., 2018).

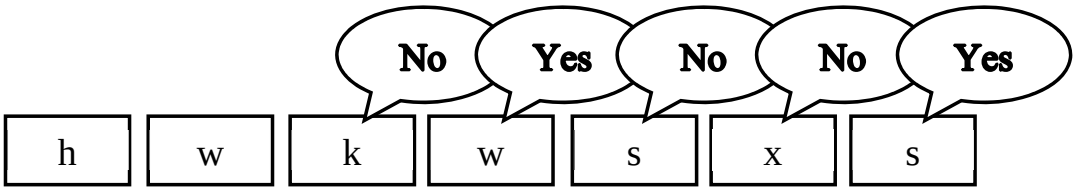


FIGURE 2.5: N-back task. A long list of items is presented to the participant. For each newly presented items, the participant must say if it was the same than the one he or she received n step back. In this example $n = 2$. Figure extracted from (Oberauer et al., 2018).

To highlight the inner manipulation of information stored, instead of having to be recalled in some way, each stimulus can also indicate how previous memory should

be updated. For instance, in the memory updating task (MU, see Figure 2.6) (Oberauer and Kliegl, 2001), a first stimulus provides an initial state (e.g. numbers) for the memory and then all the next stimulus describes the operation to transform the memory (e.g. adding 2 to the current memory). The exact processing to perform can be left to the participant to decide. For instance, there are a plethora of arithmetic tasks (e.g. addition/subtraction) that require working memory to be used (Camos, 2018; Barrouillet, 2018). Moreover, the information that subjects are asked to maintain can be more abstract in nature. For instance, in the Wisconsin card sorting test (WCST, see Figure 2.7) (Monchi et al., 2001), the information that is expected to be found and implicitly maintained in working memory is a rule to apply.

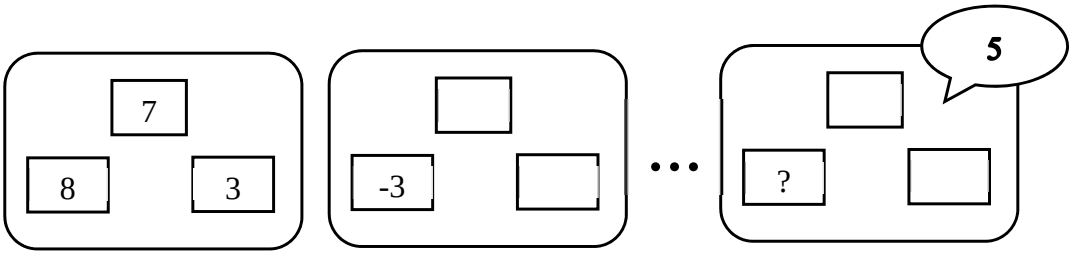


FIGURE 2.6: Memory update task. A few numbers are displayed in boxes. Intuitively, each one of these boxes represents a memory store. Afterward, a sequence of operations (e.g adding or subtracting a value) will then be individually applied to update each of these memory stores. In the end, the participant is asked what is the final value in each box. Figure extracted from (Oberauer et al., 2018).

To find the precise neuronal mechanisms involved in working memory, most of neuroscience experimental studies focus on tasks where only one single stimulus should be maintained. Whereas to find the limit of working memory, other studies experiment with much more complex tasks. Even though the task and aim of the study might vary, the first element that is considered across studies is the performance of the people/animal in the task. To have meaningful recordings, even neuroscientists need to know that in some context the animal is performing the task. Generally speaking, in most tasks, it is possible to consider a frequency of success. However, in some cases as in the continuous reproduction task, it is only possible to record the precision in the answer, i.e. the distance between the answer given and the answer expected. Moreover, other behavioral measures can be of general interest. For instance, the reaction time, i.e. the time the subject/animal takes to respond, can represent a mental load or a difficulty to perform the task. Various techniques, both invasive and non-invasive, both in humans and in animals, have been used to understand the brain's implementation of working memory. Actually, thanks to the development of non-invasive techniques, more and more studies combine behavioral results with brain recordings (Chai, Hamid, and Abdullah, 2018). For instance, functional magnetic resonance imaging (fMRI) or functional near-infrared spectroscopy (fNIRS) can allow identifying brain regions that are active while doing a working memory task. More technically speaking, fMRI and fNIRS can measure a variation in blood flow properties, such as oxygenation rate, that relates to the activity of neurons. Intuitively, when neurons are more active they require more oxygen, and thus have more oxygen supply than inactive neurons. This makes it possible to identify which regions are more active while performing a working memory task, and therefore potentially the one that plays a role in working memory. However, fMRI and

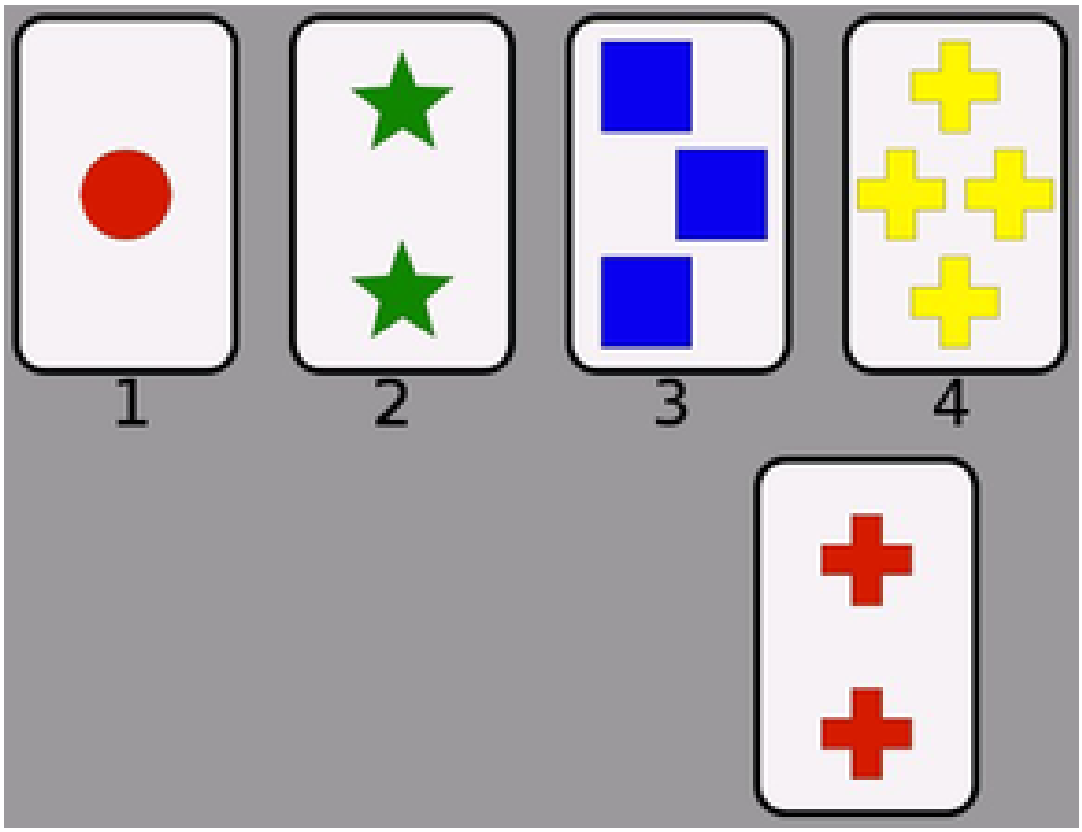


FIGURE 2.7: Wisconsin card sorting test task. Four reference cards are shown to the participant. Each reference card contains a unique type of shape (circle, star, square, plus), a unique color (red, green, blue, yellow), and a unique number of repetition of the shape (1, 2, 3, 4). The participant is repeatedly shown a new card that mixes the different attributes of the reference cards and he or she must decide with which reference card it should be associated. The participant is not aware of it but a simple rule determines what is the right reference card to choose (i.e. the one with the same shapes, color, or number of shapes). Moreover, this rule is going to change at some point during the experiment. In order for the participant to discover the rule, or to realize that it has changed, each time the participant make a choice, and he or she discovers if he or she was right or wrong. Figure extracted from (Wikipedia contributors, 2020).

fNIRS are indirect measures of what neurons are doing. In order to get closer to neuron activity, two other non-invasive technique can be used: Electroencephalography (EEG) and Magnetoencephalography (MEG). They allow recording respectively the electric and the magnetic field generated by the brain while performing working memory tasks. As, when a neuron is active it generates a local electromagnetic field, the overall electric or magnetic field that is recorded in some points can be seen as a weighted sum of the contribution of some/all neurons. In a way, EEG and MEG allow accessing to a population level activity of the neurons. To get a precise understanding of how neurons implement working memory, more invasive techniques have to be used. For instance, using implanted electrodes it is possible to record the electric field generated by a small group of neurons, or even a single neuron.

2.3 Behavioral evidences

On the behavioral side, a lot of studies are interested in characterizing the properties of working memory, i.e. to find out what are the limitations in terms of capacity, and possibly what are the influencing factors (e.g. aging). The first well-known property of working memory is that it has a limited capacity. It is not possible to hold an infinite number of items simultaneously, only a few can be. This phenomenon is known as the set-size effect, where the term "set-size" refers precisely to the number of items to be stored. The effect can be summarized by the following sentence: the bigger the set-size the worse the accuracy of recall. In other words, the more items hold simultaneously the harder it is to maintain them. That is precisely what has led to consider an upper bound on the number of items that can be held simultaneously. This set-size effect has been highlighted by many different studies on various different tasks (Oberauer et al., 2018). However, the upper bound of the number of items that can be stored is still being debated (Cowan, 2010). The first bound proposed on the number of items possibly held simultaneously was the "magical" number seven (plus or minus two) (Miller, 1956). However later, Cowan proposed that instead of having a bound on the number of items stored simultaneously, there might be a bound on the number of chunks that can be held simultaneously (Cowan, 2010). Four chunks could be held simultaneously and each chunk could possibly store several items. Moreover, the number of items held simultaneously does not only impair accuracy, but it also has an impact on the reaction time to recall. The bigger the set-size, the higher the latency. The accuracy of recall is also not the same for all items. More specifically, the order in which the items to be stored is received influences the quality of their storage. This effect is known as the serial position effect. Surprisingly, the order in which the items must be recalled does not monotonously determine the quality of their storage. Both a primacy and a recency effect can be observed, i.e. the items presented at the start and at the end of the list are better stored than the one in the middle (Madigna, 1971). Furthermore, distractions can influence the accuracy of recall. If before the recall, the subject is asked to do a task to distract him, the recall becomes less and less good the longer the distraction last (Brown, 1958; Peterson and Peterson, 1959). However, in a dual-task paradigm, Baddeley shows that if the distraction uses different "modality", then the distraction is not completely impairing the accuracy of recall, but still impairing the time to recall (Baddeley and Hitch, 1974). In addition, items seem not to be encoded in a precise way in working memory seems. The proximity of items influences how well they are stored in working memory. For instance, the closer phonologically the items to be memorized are, the more difficult it is for subjects to recall them accurately (Conrad and Hull, 1964).

2.4 Neural evidences

Even though behavioral studies allow us to better characterize the properties of working memory, it does not say anything about how the brain implements working memory. Other studies precisely aim at discovering the neural mechanisms implied in working memory. However, the functioning of working memory in the brain is far from being fully understood. Soon after the early proposal of the term working memory in the behavioral literature (Miller, Galanter, and Pribram, 1960), the prefrontal cortex (PFC) has been acknowledged to be an important region supporting

working memory (Pribram et al., 1964). In particular, it has been shown that lesions in the PFC can cause short-term memory deficits (Pribram et al., 1964).

The understanding of the role of PFC in working memory has dramatically evolved since 1960 (see (D'Esposito and Postle, 2015) for a more extensive historical overview). In the beginning, the PFC was thought to be the place where the information is maintained, the primary reason being the early discovery of sustained activity in the prefrontal cortex. To be more precise, around the same time, Fuster and Alexander (in a delayed response task) and Kubota and Niki (in a delayed alternation task) found PFC neurons with persistent firing during the delay (i.e. the time interval during which a memory must be retained) coding the memory being maintained (Fuster and Alexander, 1971; Kubota and Niki, 1971). Using fMRI, it was later demonstrated that human PFC also exhibits the same property (Courtney et al., 1997). However, as reminded by Constantinidis et al., persistent does not mean perfectly stationary, the neural activity can vary during the delay (Constantinidis et al., 2018a). In fact, the neural activity seems not to be stationary but dynamic(Stokes, 2015a). It has been hypothesized that this may be caused by a transfer of neural activities to the properties of synapses, for example using short-term synaptic plasticity even though there is no empirical evidence yet to show this. However, in some cases, it has been shown that even though dynamically encoded in the neurons, the information maintained can still be decoded from a stable readout (Constantinidis et al., 2018a). Moreover, even if experimentally observed, it is still unclear whether persistent firing of neurons is essential for maintaining (D'Esposito and Postle, 2015). On the one hand, reversible inactivation of the PFC through cooling diminish both persistent firing and working memory performance (Chafee and Goldman-Rakic, 2000). On the other hand, when the duration of the maintenance (i.e. the delay) is fixed this persistent activity might appear only late in the delay (Watanabe and Funahashi, 2007). Moreover, using fMRI (Lewis-Peacock et al., 2012) and EEG (LaRocque et al., 2013) in humans, researchers have shown that only the item in the focus of attention was represented by persistent activity, i.e. items maintained by working memory but not in the focus of attention are not represented by the persistent activity. Lundqvist, Herman, and Miller even argue that this persistent firing comes because the data recorded are averaged across time and across trials (Lundqvist, Herman, and Miller, 2018). When looking at single trials, the activity is too sparse and transient to be called persistent (Lundqvist, Herman, and Miller, 2018).

Neural activity related to information that is maintained has actually been found in many other regions, not only the PFC, or not even only the cortex but also the subcortical region (see (Christophel et al., 2017) for an extensive and recent overview). D'Esposito and Postle argues that unlike vision or motor control, working memory is not performed by a limited number of brain regions (D'Esposito and Postle, 2015). Working memory would emerge from the collective behavior of several systems (e.g. visual or motor system) aiming at reaching behavioral goals. However, PFC lesion (Pribram et al., 1964) and reversible inactivation (Chafee and Goldman-Rakic, 2000) suggests that PFC play a central role in working memory. If each region of the cortex seems to be specialized to maintain a certain type of information (Christophel et al., 2017), PFC seems to maintain a mixture of multiple task-related variables (Rigotti et al., 2013a). In particular, it seems to represent rules and goals (D'Esposito and Postle, 2015). The increasingly acknowledged hypothesis is that, using this representation of rules and goals, PFC would provide a top-down signal directing the attention (Curtis and D'Esposito, 2003; D'Esposito and Postle, 2015; Funahashi, 2017a). For instance, the direct feedback from PFC to posterior cortical region processing

sensory input could help focus on task-relevant information (D'Esposito and Postle, 2015). In fact, using a dual-task paradigm, Watanabe and Funahashi have shown that there is a big overlap in neurons maintaining information and neurons focusing the attention (Watanabe and Funahashi, 2014).

2.5 Summary

In many contexts, some information needs to be maintained for a short time. For instance, when reading, at the end of a sentence it is necessary to remember its beginning in order to understand its meaning. The memory that takes care of maintaining this information for a short period of time has had several names: primary memory, short-term memory or even working memory. The mixing of terms, their slightly different definitions, and the imperfect knowledge on their brain implementation lead to many definitions in the literature. However, the most popular term nowadays is working memory. In the modern vision, working memory is not just a memory. Working memory is conceptualized as a generic component that is responsible for both temporarily storing information and processing it. Experimental studies of working memory have had two aims: (1) understanding the properties and the limits of working memory, and (2) understanding the brain implementation of working memory. The generic aspect and the different goals lead to various different experimental paradigms. If at the behavioral side many effects have been identified, e.g. a limit to the capacity of the working memory, the brain implementation is still not perfectly known. However, the prefrontal cortex seems to play a crucial role whether by its ability to maintain information or by its ability to direct attention. Moreover, there seems to be more than one way for the neurons to maintain information (e.g. sustained activity, dynamic activity, activity silent), and these different ways could coexist.

Chapter 3

Working memory: Theoretical

The purpose of this chapter is to provide a historical perspective on working memory and to show how both definition and modelling have evolved over the past few decades.

3.1 Box and arrow model

The study of memory has a long history that dates back to the pioneer work of Richard Semon (1921, who devised the term "engram") (Semon, 1921) and Karl Lashley (1930, who hypothesized the cortical basis of learning) (Lashley, 1930). The term "working memory" as such has been introduced only later by Miller to describe what helps the execution of mental plans (Miller, Galanter, and Pribram, 1960).

One of the things that has helped to better understand memory is the characterization of the different types of memory. In the Multi Store Model of Memory (MSMM) (Atkinson and Shiffrin, 1968), memory is broken down into three different sub-systems: sensory memory, short-term memory, and long-term memory. These memories can be distinguished by their storage capacity and the duration for which their contents are kept without being lost. Sensory memory passively stores for less than a second all the stimuli the body receives (e.g vision, smells, tastes, ...). Its storage capacity is a priori very large. An attentional process selects relevant information from the sensory memory for storage in short-term memory. Only a few information can be stored simultaneously in short-term memory, up to 7 ± 2 chunks of information at a time. Short-term memory content can remain for about 15 to 30 seconds. To stay longer in short-term memory, an active rehearsal process has to be used. This rehearsal process is also the key to transfer content from short-term memory to long-term memory. Any information that has not been sufficiently rehearsed in short-term memory and transferred in long-term memory is eventually forgotten. There is no well known bound on long-term memory capacity. Long-term memory can virtually store an infinite amount of information indefinitely, which can then be retrieved in short-term memory.

In the MSMM model, transfer to long-term memory happens only through rehearsal in short-term memory. However, experimental evidence seems to highlight that rehearsal might not be the key factor to transfer information to long-term memory. The type of processing performed with the information stored in short-term memory highly influences how it is transferred to long-term memory. For instance, semantically processed words are more easily remembered than phonetically or visually

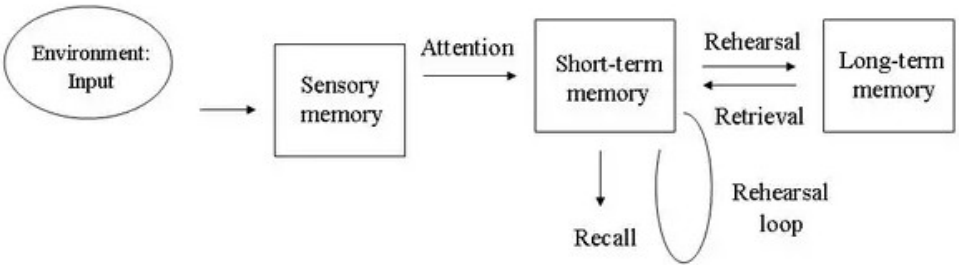


FIGURE 3.1: Memory transfer between sensory, short-term and long-term memory according to Atkinson and Shiffrin's Multi Store Model of Memory

processed words (Craik and Lockhart, 1972). This discovery revealed a profound link between memory and information processing.

Moreover, the study of a patient with short-term memory disorders but no long-term memory disorders also suggests that short-term and long-term memory might behave more independently than described in the MSMM model (Shallice and Warrington, 1970). The information transferred to long-term memory might not pass through short-term memory, and short-term memory might not be necessary for retrieval from long-term memory.

Furthermore, in the MSMM model, the short-term and long-term memory are considered as a unique storage media. But, for both of them, there is evidence showing that there are multiple storage media. For instance, long-term memory has been dissociated into episodic, procedural, and semantic memories (Tulving, 1972), and short-term memory into a visual and an acoustic part (Baddeley and Hitch, 1974).

Baddeley and Hitch were the first to introduce the term working memory to describe the system in charge of both the short-term information storage and its processing (Baddeley and Hitch, 1974). In their model of memory, short-term memory is replaced by working memory, a component that is responsible for both short-term storage and processing. This working memory is also described as multicomponent. For an extensive historical review on how their multicomponent model of working memory evolved see (Baddeley, 2012). By having an auditory serial recall and a visually presented reasoning task done simultaneously they discarded the hypothesis of a single short-term memory store. Indeed, while the response time increased slightly with the number of items to be stored, performance did not decrease. That is precisely why they considered two components: (1) the visuospatial sketchpad that stores and processes information in a visual or spatial form, and (2) the phonological loop that stores and processes information in a speech-based form. If the visuospatial sketchpad can be considered as an inner eye, the phonological loop would be composed of an inner ear, the phonological store, holding information for a few seconds, and an inner voice, the articulatory control process, which can rehearse and store in the phonological store. In their model of working memory, a system interfaces between the different other systems, the central executive. Unlike other components, the central executive is not considered as a memory store but as a component that controls attentional processes. Later on, they considered another memory store, the episodic buffer (Baddeley, 2000). It aims at storing multi-modal bindings (a.k.a. episodes), e.g. bindings between perceptions, short-term storages, and long-term memories. Some researchers have tried to bind the components of Baddeley's

model to regions in the brain (Chai, Hamid, and Abdullah, 2018; Funahashi, 2017a). Prefrontal cortex would be the central executive, Broca's and Wernicke's area the phonological loop, the occipital lobe the visuo spatial sketchpad, and the parietal lobe the episodic buffer.

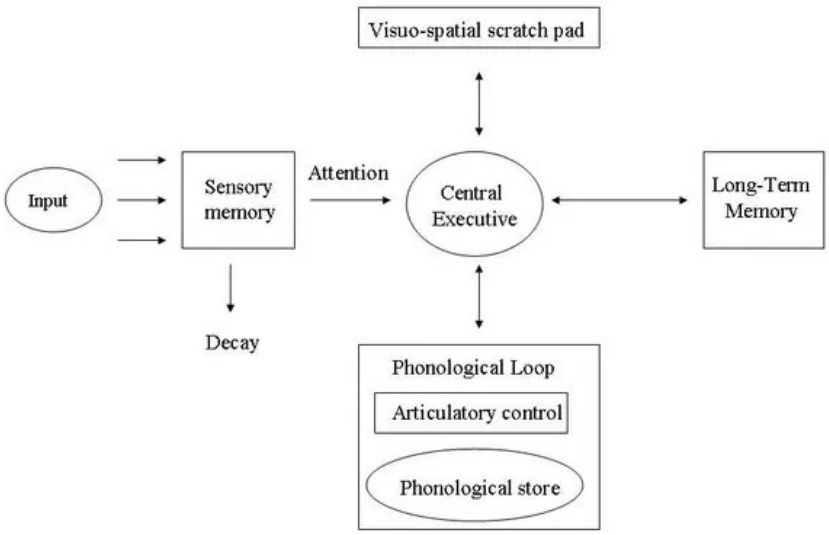


FIGURE 3.2: Baddeley's multicomponent model of working memory

w

Interestingly the triple code of numbers in the brain proposed by Dehaene shows a similar decomposition of the representations than Baddeley's model (Dehaene, 1997). Dehaene claims, that in the brain numbers have a visual form, that is what makes us able to recognize or write them. They also have a speech based form, that is what makes us able to pronounce them and to recognize when they are pronounced. But the biggest claim was that numbers have an even more primary form. Numbers represent quantities and, in the brain, the numbers are also present as quantities. However, this representation in quantitative form is not perfect. There is a distance effect, the larger the quantities, the more distant they have to be in order to be dissociated. This primary form of numbers have not been found only in humans. This form of numbers exists for instance in primates and even in young babies who can't talk yet (Dehaene, 1997).

Later, to better understand the attentional process, Cowan proposes a similar mechanism of interaction between the central executive, short-term and long-term memory (Cowan, 1999). Cowan considers short-term memory to be an activated version of long-term memory, and the central executive to direct the attention on a selected few items in short-term memory. While Baddeley's model puts forward different short-term components, Cowan's model puts forward a universal limit of the attentional focus. In Cowan's model, the capacity of working memory lies within this limit, four chunks (which may contain more than a single item) can be in the focus of attention at a time.

One of the major reasons for considering working memory in the first place was the strong link between short-term storage of information and its processing. However, neither Baddeley's model nor Cowan's model explains how this processing happens and their link with short-term memory. To fill this gap Oberauer proposes

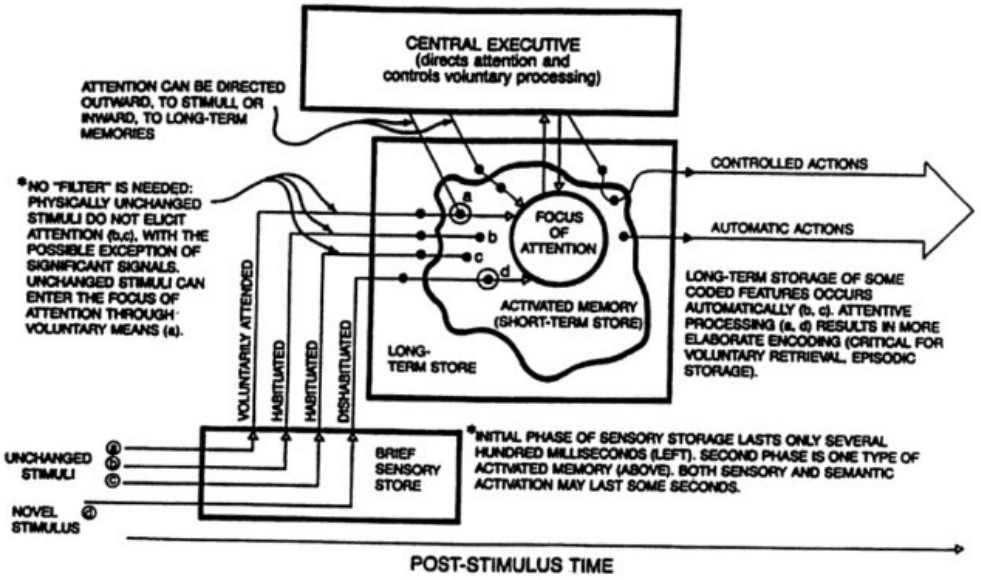


FIGURE 3.3: Cowan's model of working memory.

to consider two different types of working memory (Oberauer, 2009): (1) a declarative working memory responsible to make representations available for the processing, and (2) a procedural working memory responsible for the processing. In Obereauer's model, while there are attentional processes, there is no explicit component such as the central executive to guide them. As in Cowan's model in the declarative component, the short-term memory is first in the form of the activated long-term memories, and the focus of attention will eventually select a few of them to be processed. However, short-term memory goes through an intermediate representation that can help in the selection process. This intermediate representation is performed by the direct-access region. The direct-access region can select a few characteristic features of short-term memory or embed short-term memories into a structure. In Obereauer's model the procedural short-term memories represents potential processing to apply. They are first in the form of activated long-term memories and a response focus selects which one to apply. The bridge is what performs the processing of the selected declarative memory. It holds task-sets, i.e. reflexes binding selected declarative components to how it should be processed. Intuitively, the bridge and the direct-access region are similar in the sense that they both bind representations. Interestingly, both declarative and procedural working memory are described in a very similar way: three sub-component embedding three successive levels of representations.

3.2 Dynamical systems models

Another way to describe how working memory works is to use dynamical systems. For a more thorough review on these approaches see (Durstewitz, Seamans, and Sejnowski, 2000), (Compte, 2006) and (Barak and Tsodyks, 2014). In a dynamical system, there are many ways to represent memories. Intuitively, a memory should be something that is stable, thus the first immediate solution was to consider the attractors of dynamical systems as its memory states. Attractors are regions in the state space that are stable and resistant to local disturbances, and therefore the ideal

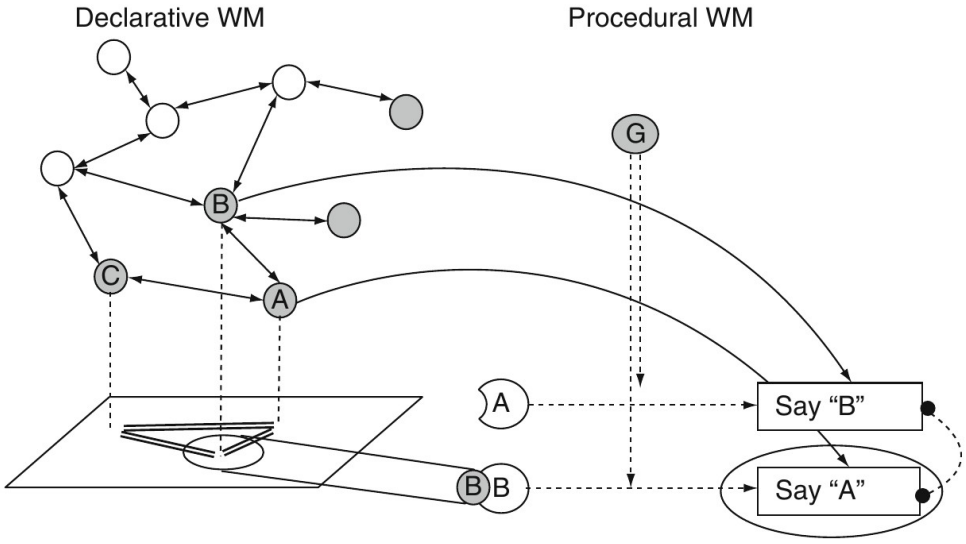


FIGURE 3.4: Obereauer's model of working memory. Left: The three levels of declarative working memory. Activated long-term memory

candidate to represent memory. Attractors are especially suited to represent short-term memories because with strong enough perturbations it is still possible to make the dynamical system switch from one attractor to another, as short-term memory would go from one state to another. A line of theoretical research has therefore tried to understand how to build these attractors.

For instance, Hopfield have shown a method to build any point attractors in a binary recurrent neural network[1] (Hopfield, 1982). The most astonishing fact is that this method is based only on a very simple Hebb's principle. "Neurons that fire together, wire together. Neurons that fire out of sync, fail to link." (Hebb, 1949) Given a binary pattern[2], Hopfield consider a binary recurrent neural network which has the same number of neurons than the dimension of the pattern. Each neurons can thus be associated to a value of the binary pattern. All the neurons are connected to other neurons but have no self feedback, and the connections between neurons associated with the same values are positive while those between different values are negative. The synaptic weights thus become characteristic of the pattern. As Hebb's principle states, in this network, opposite states repel each other while identical states attract each other. The neurons associated with 1 (resp. -1) tend to have the same value. While, the neurons associated to 1 and neurons associated to -1 tend to have oposite values. Thus, starting from a perturbed (close) version of the pattern, the network will reconstruct the pattern in its activity. In other words, the pattern became an attractor of the network. The exact same idea extends to several binary patterns, by considering the average synaptic weight characteristic of the patterns. However, the number of attractors that can be built with this method is limited by the number of units, and therefore by the size of the patterns. This way of building attractors has the advantage of building at the same time a content adressable memory.

Amari proposed one of the first ways to build attractors with plane neural fields (Amari,

[1] In this binary neural network, the activities of neurons take only the values 1 or -1.
[2] Similarly than the binary neural network, this binary pattern is a vector of values in $\{-1, 1\}$

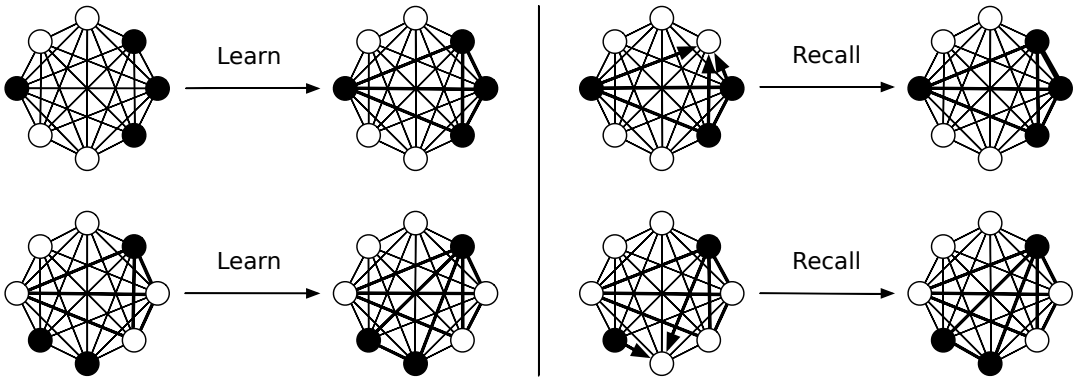


FIGURE 3.5: Hopfield network's principle. Top left: Learning patterns. The patterns are enforced one after the other in the neurons, and the weight between simultaneously active (resp. inactive, for the sake of visualisation only active in the Figure) neurons are sthrengthened, and the others are weakened. Top right: Recalling patterns. When a pattern is partially enforced in the neurons, the faithfully active (resp. inactive) neurons of the pattern will enforce the inactive (resp. active) neurons that should have been active (resp. inactive) to be active (resp. inactive).

1977). The idea was the following. If the neural field undergoes long range inhibition and short range excitation, then, when it gets locally excited in a particular point and because of the long range inhibitions, all the activity will be concentrated at this point. Moreover when the external excitation is gone, the short range excitation will make the activity to last and to be localized at that point. Each point thus becomes a potential attractor in the form of a bump. If these bump attractors were originally considered to evolve on a plane, the same idea was later used to arrange them on a ring. In fact, Compte proposed to use a ring structure formed of leaky integrate and fire neurons with short range excitation and constant inhibition (Compte, 2000). Such hypothesis, i.e. constant inhibition, still allowed the formation of the bumps attractors on the ring. Interstingly a dedicated circuit implementing these bumps on a ring have latter been experimentally found in the Drosophila (Kim et al., 2017). This idea was later extended to try to better understand the conditions under which two bumps can coexist, as two items can be simultaneously held in short-term memory (Edin et al., 2009; Wei, Wang, and Wang, 2012). Another simple way to hold multiple items simultaneously with ring attractors is simply to multiply the number of rings, each ring holding its own item. But it is well known that short-term memory has a limited capacity. Bouchacourt and Buschman have investigated precisely how making these neural rings to interact through a layer of neurons limits the number of items that can be stored simultaneously (Bouchacourt and Buschman, 2019a).

In bump attractors, what remains constant is the position of the bump. In other words the information maintained is not really contained within the dynamics of a particular neuron but rather in the whole dynamics of the group of neurons. The information being maintained simply resides in one neuron being active or not. Other models consider that, what can remain constant is the firing rate of a neuron, or of a group of neurons. For instance, Lim and Goldman proposes to use the firing rate of a neuron to build a line attractor, i.e. an attractor in the shape of a line (Lim and Goldman, 2013a). The idea is as follows. They model neurons using the following dynamics: $\tau\dot{x} = -x + u$, where x is the firing of neurons, u the input of the neurons, and τ a time constant. In this model, the information is given in the input of the

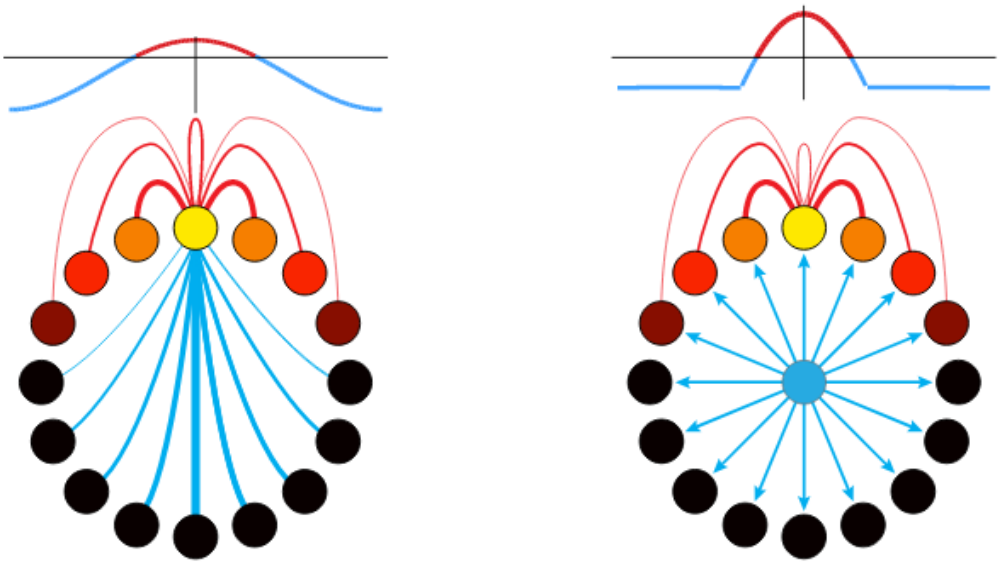


FIGURE 3.6: Two ways to build a ring attractor based on the same idea: short-range excitation, long-range inhibition. Right: with neurons that can be both inhibitory to some neurons and excitatory to others. On the ring, neurons are more excitatory to closer and more inhibitory to further neurons. Left: with only inhibitory/excitatory neurons. The ring is formed of excitatory neurons which projects stronger to closer neurons. Inhibitory interneurons are added to enforce most neurons to remain inactive

neuron. More precisely, when the neuron receives a non-zero input its firing rates quickly match its input. Afterward, the firing rate of the neuron gradually vanish. τ is thus what directly influence the speed in which the information is lost, the bigger τ the longer the information lasts. What makes these neurons naturally forgetful is their small time constant τ, i.e. τ is in the millisecond range. In order to keep the information longer it is possible to use an excitatory feedback. However in order to be precise enough, this feedback should be very finely tuned. To increase the duration the information remains without fine tuning Lim and Goldman makes the following remark. If a neuron ever receives negative derivative feedback then it directly influences its time constant, allowing then to artificially increase the duration the information remains. They show how in practice it requires less precise tuning, and how, combined with an excitatory feedback, it is still possible to maintain information for a longer period at the neuron level. They also show how such negative derivative feedback can be obtained by the interaction of two populations of excitatory and inhibitory neurons evolving at different time scales.

In the different models presented previously, whether stable fixed points, bump attractors in the form of plane or ring, and even line attractor, a memory state is always associated with one single state of the network. However there are alternative ways to implement memory, such as for instance using a periodic attractor to represent a memory. Intuitively, a periodic attractor is a stable region that is resistant to perturbations in which the dynamic is periodic. Models such as conceptors allow to build such periodical attractors (Jaeger, 2017a). To build a periodic attractor, Jaeger

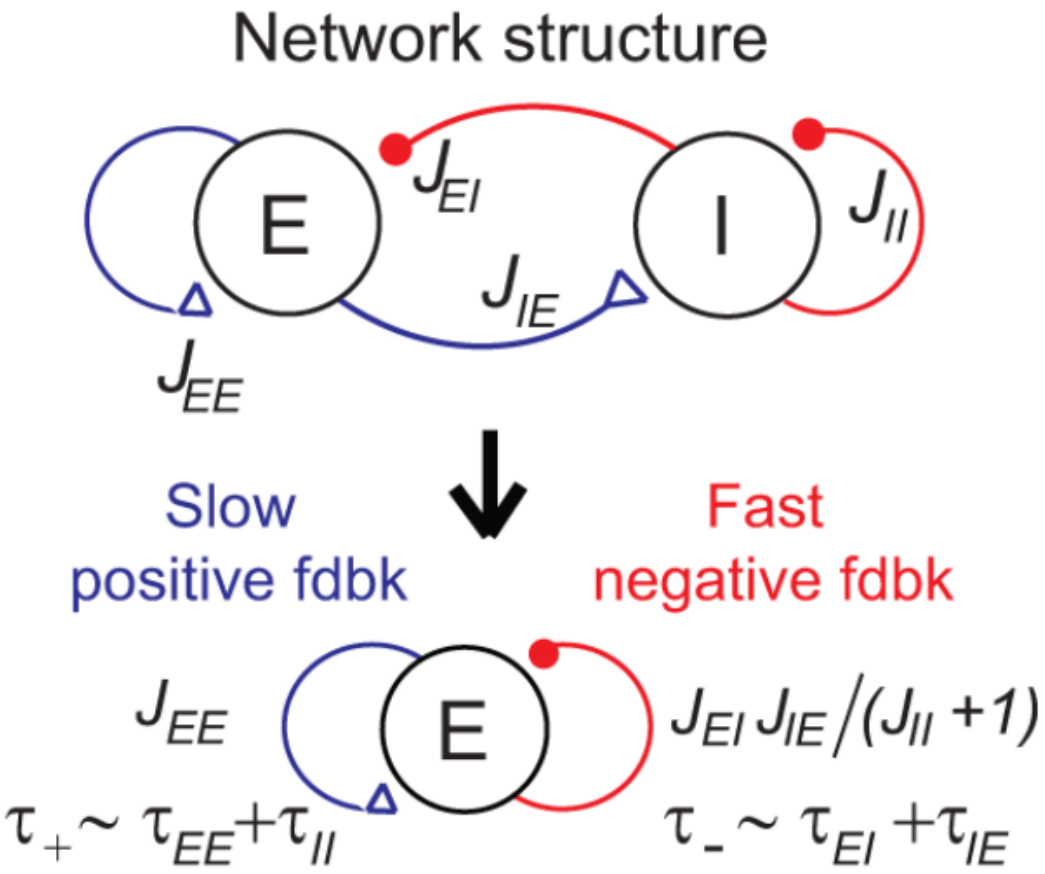


FIGURE 3.7: Lim and Goldman's negative derivative feedback implementation. (Top) The network consists of one population of excitatory neurons, and one population of inhibitory neurons. (Bottom) From the excitatory population point of view, everything happens as if it received both a positive feedback and a negative feedback. They show how a negative derivative feedback emerge from the superimposition of a slow positive feedback and a fast negative feedback of equal strength.

proposes to feed a network with a periodical input. Indeed, under certain conditions[3], it is expected that when a dynamical system receives a periodic input it will tend exponentially and rapidly towards a unique periodic trajectory. This trajectory is precisely the candidate Jaeger chose to form the periodic attractor. In practice, he makes learn the network learn to produce this trajectory without the input. To build several periodic attractors he proposes to learn only one trajectory, the average between the different trajectories. By noting that the trajectories obtained by giving different periodic inputs can potentai:lly evolve in different spaces, Jaeger proposes then to retrieve one or the other trajectory by continuously projecting the dynamics of the average trajectory in the space of the selected trajectory.

If the classical notion of attractor makes it possible to represent the memory of a dynamical system which receives sparse inputs, when these inputs are no longer sparse it is no longer sufficient. This is why Pascanu and Jaeger introduced the concept of input induced attractor (Pascanu and Jaeger, 2011a). An input induced

[3] something like the norm of the Jacobian is strictly smaller than 1 everywhere)

attractor is defined very similarly from a classical attractor. The difference is that an input induced attractor must not only be stable to local disturbances, but given a description of eligible inputs, it must be stable to local disturbances regardless of the eligible input received.

In all the models mentioned above, the notion of attractor is the key point. Most of these models propose ways to build attractors. But the very strong stability of an attractor might deserve it. In fact, as long as there are no perturbation, by virtue of being an attractor, the information actively encoded in the attractor cannot be lost. However, in short-term memory, the memory do not have to and might not be maintained forever. Regions of the state space that are not stable but in which activity remains for a certain time are also a good candidate. For example, it is possible to have such candidates in delay lines or synfire chains (Abeles, 2012). The principle of the two is similar, a chain of neurons connected in a feedforward way, the information enters in the beginning of the chain, is transmitted along the chain and then is lost at the end of the chain.

Most models attemps to build attractors in the spiking or firing rate neural activity. However as shown in Chapter 2 some information might not be maintained directly in the neural activty. Mongillo, Barak, and Tsodyks proposed that some information can be stored for a short time in synaptic properties as well (Mongillo, Barak, and Tsodyks, 2008a).

Most of the theories outlined above focus on how information can be maintained in the brain and not on how the brain decides what information should be maintained. In order to explain also how the brain decides what information to maintain O'Reilly proposes a way to implement a gating in the brain (O'Reilly and Frank, 2006a). In this model, basal ganglia, a brain region known for its crucial role in decision making, plays the role of the gate which controls what information enters and leaves the frontal cortex. He describes then how brains could act as an actor critic which learns when to gate information in the frontal cortex.

3.3 Finding WM mechanisms by emergence

Most of the dynamical system models of working memory mentionned above are handcrafted. Some parameters such as weight are chosen in a smart way so that a behavior maintaining information arises. Another strategy has been used in order to find mechanisms maintaining information without having to handcraft them. The idea is to use recurrent neural networks (RNN) and to train them to perform working memory tasks.

For instance, in the early 90s, Zipser et al. have trained a RNN to perform a gating task (Zipser et al., 1993). In this task, the RNN receives two inputs, one containing information, the other stating when the information should be loaded in. Then, by transforming the firing rates of the neurons into spikes they could find neural representation in the RNN very similar to the one that have been recorded in the brain in several different studies. In a way, it shows how important this gating principle is.

More recently, in order to investigate the possibility of synaptic encoding of working memory, Masse et al. train an RNN submitted to short-term synaptic plasticity (STSP) to do several working memory tasks (Masse et al., 2019a). They show that

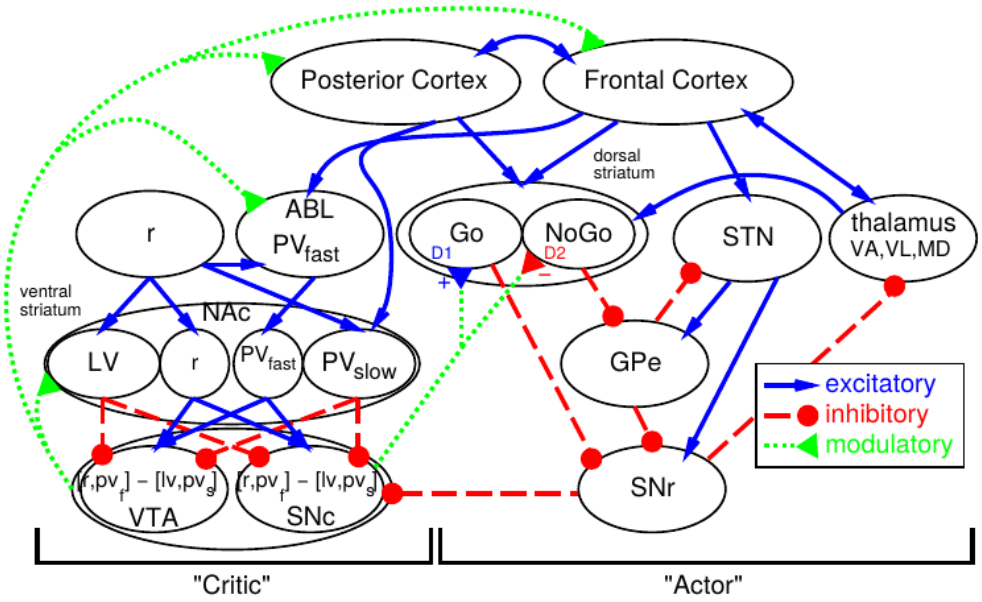


FIGURE 3.8: Summary of the PBWM model of O'Reilly and Frank. The model is composed by an actor and a critic. The actor implements the gating and the critic learns when to gate.

even if thanks to the STSP the memory can be decoded from the synaptic weights, persistent activity still arises naturally when manipulation of information is required.

Yang et al. trained an RNN to perform multiple different tasks that do not depend only on working memory but also on decision making or inhibitory control (Yang et al., 2019). They have shown that after training, it is possible to identify in the RNN functional clusters of neurons specialized in the different cognitive aspect of the tasks (e.g. working memory or decision making).

Overall, once these RNN are able to perform a working memory task, the only thing left to do to propose a mechanism is to figure out how the RNN does maintain. Barak argues that handcrafted model usually use low dimensional intuitions, and that trained RNN also end up implementing low dimensional intuitions (Barak, 2017). Recently, more and more theoretical work especially aims at discovering what are the limits of the low dimensional solution of RNNs, and how to obtain them (Schuessler et al., 2020a).

3.4 Summary

In this chapter, we have seen how there have been two different approach to model working memory: (1) using box and arrow models and (2) using dynamical systems. In a way these two approaches seek to understand/describe working memory at different levels. And ideally the aim would be to be able to combine them in order to have a multi-layered understanding of the working memory. However, in practice it is not that easy to transform box and arrow models, such as the one of Baddeley and Hitch(Baddeley and Hitch, 1974), into dynamical system model, and dynamical system models are not necessarily tightly linked to a box and arrow model. However, box and arrow models such as the one of Oberauer seems promising to make such

links more easily as the description of the model is thought along with an idea of implementation. Moreover, if many dynamical system models have been proposed to explain how the brains maintain information, there are no empirical findings that would tip the balance in favour of one of them. In practice a superimposition of all these mechanisms might take place. Most of these dynamical systems are hand-crafted but we have shown an alternative to find model using recurrent neural networks trained to perform working memory tasks.

Chapter 4

Recurrent neural networks

As mentioned earlier in this thesis, Recurrent Neural Networks (RNNs) can be used to uncover possible mechanisms underlying working memory. The procedure is as follows: an RNN is trained to solve a working memory task and then an in-depth analysis of the RNN reveals what are the mechanisms supporting working memory. This chapter aims to give an introductory overview of RNNs and why solving working memory tasks is hard. To do so, we will start by introducing sequential supervised learning tasks, and explain that solving them implies to learn how to extract contextual information (see Section 4.1). The focus will then switch on Simple Recurrent Networks, neural networks that have been designed to learn to extract context (see Section 4.2), and on the difficulties they face to extract context distant in time (see Section 4.3). Finally, we are going to present three different solutions that have been proposed to overcome these problems: (1) the Reservoir Computing (RC) paradigm (see Section 4.4), (2) using memory cells instead of neurons such as in Long Short Term Memory networks (LSTMs) (see Section 4.4.1), and (3) using explicit external memory mechanism as in Neural Turing Machines (NTMs) (see Section 4.5).

4.1 Supervised sequential learning and time dependencies

From a broad perspective, we can consider that the aim of a supervised learning task is to approximate a function f known for only a few points. Formally, a supervised learning task can be defined as a set E of pairs (x, y) such that $y = f(x)$. x and y are going to be called respectively the input and the desired output. In sequential supervised learning either the domain of f, or the codomain of f, or both are finite sequences of variable length. For instance, when the aim is to classify time series in two classes, the domain of f could be $\mathbb{R}^* = \bigcup_{k=0}^{+\infty} \mathbb{R}^k$, and the codomain of f could be $\{0, 1\}$. In the context of this thesis, both the domain and the codomain of f are finite sequences of variable length. This is the case, for example, for prediction tasks. Moreover, we assume that for all x in the domain of f, x and $y = f(x)$ are sequences of same length. In that case, we can define the time dependency between x and y as the minimal length required to map contiguous subsequence of x to their associated subsequence in y. Formally we can define $TD(x, y)$, the time dependency between x and y, and similarly $TD(E)$, the time dependency of a supervised learning task E

as:

$$TD(x,y) = \min\left\{k/\forall n,n'\ x[n:n+k] = x[n':n'+k] \Rightarrow y[n:n+k] = y[n':n'+k]\right\}$$
$$TD(E) = \min\left\{k \;\middle/\; \begin{array}{l}\forall (x,y),(x',y') \in E \\ \forall n,n'\ x[n:n+k] = x'[n':n'+k] \Rightarrow y[n:n+k] = y'[n':n'+k]\end{array}\right\}$$

where $a[p:q]$ stands for the tuple $(a[p], a[p+1], ..., a[q])$.

To familiarize ourselves with this notion of time dependency, let us look at two examples of tasks. Let us consider the periodic sequence $x = [1,2,3,2,1,2,3,...]$, and two sets E and E' corresponding respectively to the prediction of x one step ahead, and two steps ahead. Let us denote as y and z respectively the one step and two steps prediction of x, i.e. $y = [2,3,2,1,2,3,2...]$ and $z = [3,2,1,2,3,2,1...]$. Formally we can write $E = \{(x,y)\}$ and $E' = \{(x,z)\}$ (see Figure 4.1 for a representation of the task and of the time dependencies). We can start by noticing that $TD(E') = 0$. Indeed, for all n, given only $x[n]$ it is possible to predict $x[n+2]$. Two steps after a 1 there is always a 3, two steps after a 3 there is always a 1, and two steps after a 2 there is always a 2. Moreover we can also notice that $TD(E) > 0$. If 1 and 3 are always followed by a 2, without contextual information, it is impossible to predict the successor of 2 (1 or 3). However, with a one time step contextual information, it becomes possible to predict the successor of 2. If 2 was preceded by a 1 (resp. 3) then the next will be a 3 (resp. 1). In other words, for all n, given only $x[n-1:n]$ it is possible to predict $x[n+1]$. That is precisely why $TD(E) = 1$. Literature on experience related to working memory provides many examples where the time dependency can be as big as one wants. Typically, when a sequence of stimuli is given and the answer after the stimuli depends on the first stimulus, the time dependency is the number of stimuli. Indeed, the first stimulus is the contextual information that has to be remembered after all stimuli have been received. This shows how the time dependency is directly linked to the distance in time of the contextual information required. The further backward in time the contextual information is, the bigger the time dependency will be.

This contextual information is actually what links working memory and time dependency. Working memory is the memory that has to remember the contextual information. However, time dependency is not directly linked to a working memory load. To see more in-depth the link between working memory and time dependency let us consider an example of a task from the working memory experimental literature: the n-back task. In this task, the subject is presented with a sequence of stimuli and must indicate when the current stimulus matches that of the previous n stimulus in the sequence. To perform the task the subject has to keep remembering and updating the last n stimuli he has received. This is the property we are going to focus on. To simplify let us consider that instead of comparing the current stimulus with another stimulus, the participant has to recall the last n stimuli he saw each time he receives a new stimulus. That is precisely what has been done in the literature to measure what is called the memory capacity of an RNN (Jaeger, 2002). Let us model this task in two ways: (1) the null stimulus is considered to be a stimulus to remember (as in (Jaeger, 2002)), (2) it is not considered to be a stimulus to remember (see Figure 4.2 for a representation of the two tasks and of the time dependencies). In the first case, performing the tasks means producing at each time step the last n inputs that have been received, and the temporal dependency is n. Whereas in the second case it requires to produce the last n non-zero inputs received, and the temporal dependency is infinite. Indeed, the last n non-zero inputs received can be

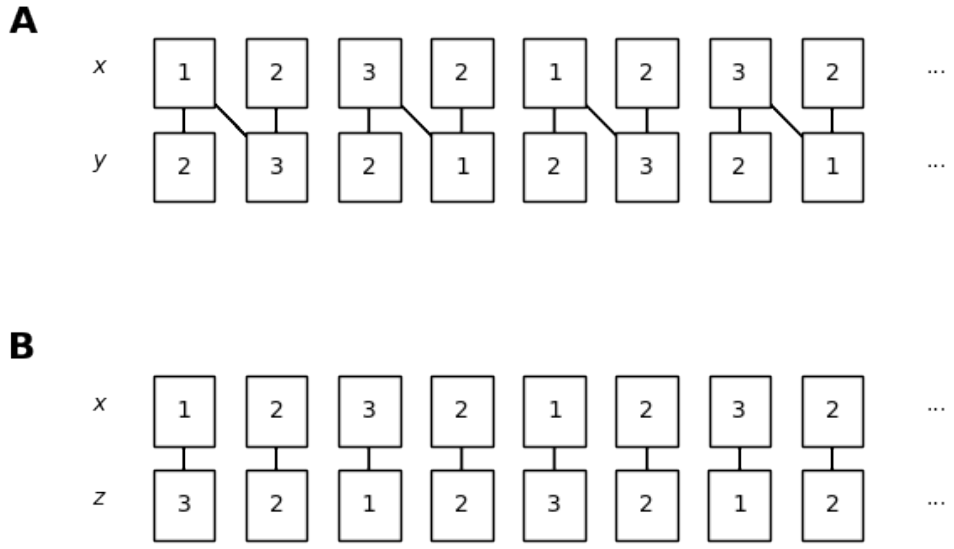


FIGURE 4.1: Prediction of periodic sequence $x = [1, 2, 3, 2, 1, 2, 3, ...]$. Lines linking blocks highlights the dependencies between x and y. **A**. One step ahead. **B**. Two step ahead.

arbitrarily far in the past. However, in both cases, only n contextual information has to be remembered. The time dependency is rather linked to how long the contextual information is expected to be maintained, rather than to the number of contextual information that has to be maintained.

One way to explicitly deal with time dependency is to spatialize the time. Indeed, it is possible to unfold the time of the input by considering the following dynamics:

$$c[n] = [x[n], c[n-1][:-1]])$$

where x is the input, c is the unfolded version of the input, $[a, b]$ represents the concatenation of a and b, and $a[:-1]$ represents the removal of the last component of a. However, such unfolding have three main drawbacks: (1) in practice it increase dramatically the dimensionality of the input, (2) the number of previous times step that should be included (i.e. the dimension of c) has to be manually set, and (3) it does not allow to deal with infinite time dependency. As we have seen earlier, even remembering a single contextual information in working memory can lead to an infinite time dependency. In this case, extracting and maintaining the contextual information have somehow to be learned.

Most of the time in supervised learning tasks, the function f is unknown, and the approximation of f will be used to infer new values of f, i.e. to create new potential pairs (x', y') such that $y' \approx f(x')$. In the context of this thesis, in some cases, f is

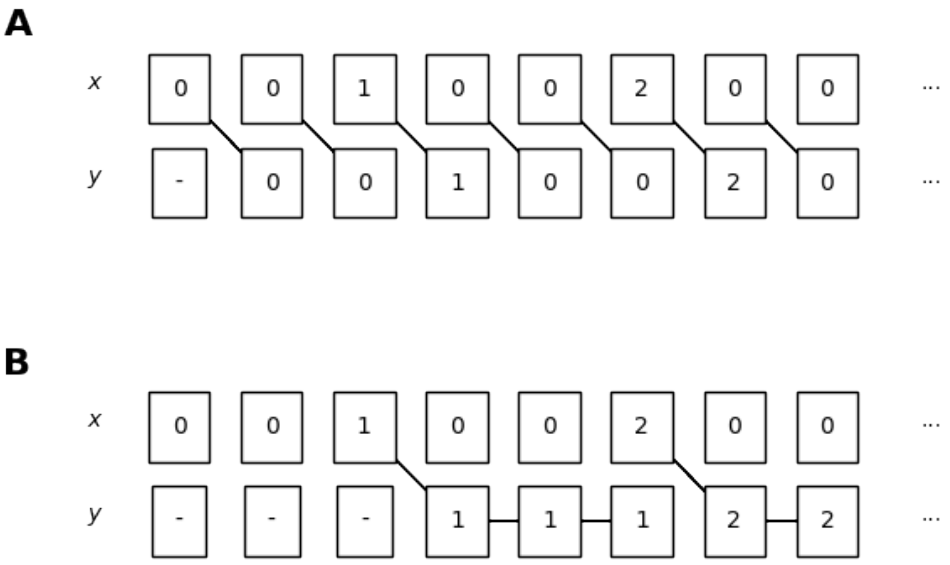


FIGURE 4.2: Simplification of the 1-back task. **A**. All stimuli should be remembered. **B**. Only non-zero stimuli should be remembered.

known. In that case, the interest for us is not to predict new values for f but to find out what mechanisms are used to approximate f.

4.2 Simple Recurrent Networks

In order to build an Artificial Neural Network (ANN) which can learn to extract the context, Elman proposed to extend a multilayer perceptron (MLP) with contextual units (Elman, 1990). More precisely, he considered a MLP with one hidden layer, and added contextual units which copy the value of the previous hidden state and provide it as input to the hidden layer. A modern way to describe it, without contextual units, consists in saying that Elman has added feedback connection from the hidden states to itself. ANN with such feedback connections have been called Recurrent Neural Networks (RNNs), and Elman Network has been popularized as the Simple Recurrent Network (SRN). The SRN dynamics can be discribed as follow:

$$x[n] = f(Wx[n-1] + W_{\text{in}}u[n])$$
$$y[n] = g(W_{\text{out}}x[n])$$

where x is the RNN state sequence, u the input sequence received by the RNN and y the output sequence produced by the RNN, f and g are two functions, the internal activation function and the output activation function, and W, W_{in} and W_{out} are called respectively the recurrent weights, the input weights and the output weights.

To learn to extract the context and to perform a task W, W_{in} and W_{out} have to be optimized. The classical optimization method used for SRN is an adaptation of the backpropagation algorithm used in MLP. It is known as BackPropagation Through Time (BPTT) and it consists in unfolding time, thus transforming a SRN in a MLP (Rumelhart, Hinton, and Williams, 1985). Even though SRNs can virtually approximate any open dynamical systems (Schäfer and Zimmerman, 2007) and thus handle any time dependencies, learning tasks with long time dependencies is hard in practice. The reason is that it requires to propagate an error gradient over long period of time, which tends to make it explode or vanish (Bengio, Simard, and Frasconi, 1994).

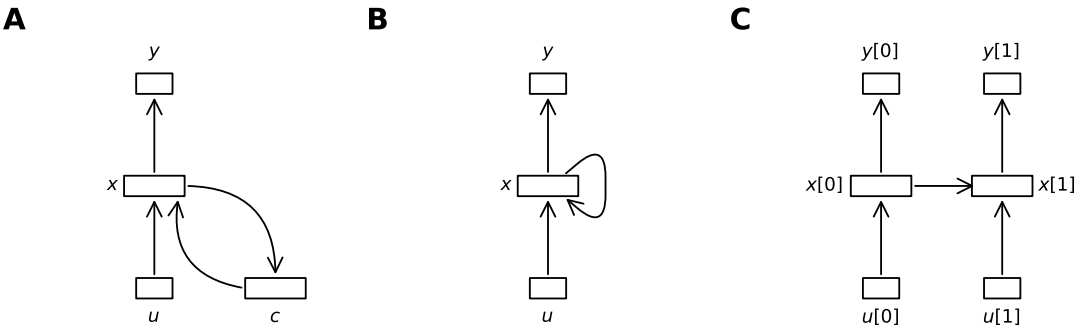


FIGURE 4.3: Simple Recurrent Network (SRN). **A**. Initial proposal of Elman (Elman, 1990). **B**. Equivalent with feedback connections from the hidden layer to the hidden layer. **C**. Unfolding in time transforming the SRN in a MLP. u input, x hidden recurrent state, y output, c contextual unit.

In Elman Network, the contextual units contains information about the hidden states. Jordan proposed a variant where that contextual units contains information about the output instead (Jordan, 1997). In a sense it's like adding feedback from the output to the hidden layer instead of feedback from the hidden layer to itself. The Jordan Network dynamics can be described as follows:

$$x[n] = f(W_{\text{fb}} y[n-1] + W_{\text{in}} u[n])$$
$$y[n] = g(W_{\text{out}} x[n])$$

where W_{fb} is called the feedback weights.

This idea to provide contextual units (or feedback) has been widely used ever since. For instance variants combining Elman's and Jordan's ideas have been used (Pham and Karaboga, 1999). These ANNs are called recurrent because in these networks some neurons send feedback signals to some other neurons, which creates recurrent loops in the network. If feedforward networks such as MLPs implement functions, RNNs implement open dynamical systems. More precisely, RNNs are open dynamical systems, i.e. they evolve in time and their dynamics are governed by the inputs they receive. Their dynamics can be described as follows:

$$x[n] = f(x[n-1], u[n])$$
$$y[n] = g(x[n])$$

where x is the RNN states sequence, u the input sequence received by the RNN and y the output sequence produced by the RNN, and f and g are two functions. Intuitively, f describes the flow of the RNN, i.e. how the states will evolve given an input sequence. Whereas g describes how to extract a meaningful and easily

interpretable information from the states, a.k.a. its output. It is f who is in charge of extracting the necessary contextual information from the input.

4.3 Fading memory and vanishing gradient

Let us explain in more details the exploding/vanishing gradient problem and its influence on learning task with long-term time dependencies. For the sake of simplicity, let us consider the case where there is only one neuron and the internal activation function is the identity function. In that case the dynamics of the SRN is written as follows:

$$x[n] = wx[n-1] + w_{\text{in}}u[n] \tag{4.1}$$

where x, u, w and w_{in} are scalar values.

Moreover, let us suppose that $x[n]$ has $d[n]$ as a target, we can thus define an error $e[n]$ as:

$$e[n] = (x[n] - d[n])^2 \tag{4.2}$$

First of all, if we solve the recursion, we can rewrite Equation 4.1 as follows:

$$x[n] = \sum_{k=0}^{n} w^{n-k} w_{\text{in}} u[k] \tag{4.3}$$

In Equation 4.3, we can notice the exponential factor w^{n-k}. When $|w| > 1$, as $|w|^n$ tends towards infinity, $x[n]$ will become less and less dependant from $u[k]$ the closer n is from k. In fact when $|w| > 1$, the dynamics evolves in a chaotic way, a slight perturbation in u will be exponentially amplified. Similarly, when $|w| < 1$, as w^{n-k} tends towards 0, $x[n]$ will become less and less dependant from $u[k]$ the further n is from k. However, the closer $|w|$ is to 1, the more $u[k]$ in the past, $x[n]$ will depend on. Eventually when $|w| = 1$, $x[n]$ will depend in the same way from all $u[k]$. In fact, this result is more general, there is a frontier beyond which a RNN evolve chaotically, and below which it will have a fading memory of the input.

Let us show the influence of this exponetial factor on learning. To learn by gradient descent, the partial derivatives $\frac{\partial e[n]}{\partial w}$ are used to update w. More precisely, it is the sum of the $\frac{\partial e[n]}{\partial w}$ that will be used to update w. Using Equation 4.3) it is easy to compute these partial derivatives:

$$\frac{\partial e[n]}{\partial w} = \frac{\partial e[n]}{\partial x[n]} \frac{\partial x[n]}{\partial w} \tag{4.4}$$

$$= 2(x[n] - d[n]) w_{\text{in}} \sum_{k=0}^{n-1} (n-k) w^{n-k-1} u[k] \tag{4.5}$$

Similarly than in Equation 4.3, in Equation 4.5 there is an exponential factor w^{n-k-1}. The same conclusion can thus be made for $\frac{\partial e[n]}{\partial w}$ than for $x[n]$. If $|w| < 1$, $\frac{\partial e[n]}{\partial w}$ depends mostly on the last value of u, whereas if $|w| > 1$, $\frac{\partial e[n]}{\partial w}$ depends mostly on the first

value of u. Moreover, when $|w| > 1$, it is possible that $\frac{\partial e[n]}{\partial w}$ grows exponentially with n, thus causing a very big change in one single learning step. To avoid such problems, Pascanu, Mikolov, and Bengio proposed to clip the norm of $\frac{\partial e[n]}{\partial w}$(Pascanu, Mikolov, and Bengio, 2012). Furthermore, for sufficiently large n, when $|w| < 1$, $u[k]$ will have disappeared from $\frac{\partial e[n]}{\partial w}$. The change in w caused by learning will therefore be blind to the $u[k]$ distant in time. Intuitively, it makes it difficult to make $x[n]$ depend on the $u[k]$ distant in time.

In practice, it is possible to reconsider this exponential factor in a way that is easier to generalize to the non linear case. To do so, let us unfold the time for w by making it artificially dependant on time.

$$x[n] = w[n]x[n-1] + w_{\text{in}}u[n] \tag{4.6}$$

Now let us compute all $\frac{\partial e[n]}{\partial w[n-k]}$. To do so, we can use the partial derivatives in relation to the hidden states. By iterating the chain rule, we can write:

$$\frac{\partial e[n]}{\partial x[n-k]} = \frac{\partial e[n]}{\partial x[n]} \prod_{i=1}^{k} \frac{\partial x[n-k+1]}{\partial x[n-k]}$$

This brings to light the partial derivatives $\frac{\partial x[n-k+1]}{\partial x[n-k]}$, and using Equation 4.6 we can note that it has a very simple form:

$$\frac{\partial x[n-k+1]}{\partial x[n-k]} = w[n-k+1] = w$$

It is thus possible to obtain the partial derivative $\frac{\partial e[n]}{\partial x[n-k]}$ as follows:

$$\frac{\partial e[n]}{\partial x[n-k]} = \frac{\partial e[n]}{\partial x[n]} w^k$$

Here appears the exponential factor again. When n grows, $\frac{\partial e[n]}{\partial x[n-k]}$ either shrinks to zero when $|w| < 1$, or explode when $|w| > 1$. In order to avoid such explosion, Graves proposed to clip the norm of $\frac{\partial e[n]}{\partial x[n-k]}$ (Graves, 2013). This has a direct effect on $\frac{\partial e[n]}{\partial w[n-k]}$ because:

$$\begin{aligned}\frac{\partial e[n]}{\partial w[n-k]} &= \frac{\partial e[n]}{\partial x[n-k]} \frac{\partial x[n-k]}{\partial w[n-k]} \\ &= 2(x[n] - d[n]) w^k x[n-k-1]\end{aligned}$$

Unsurprisingly, using again Equation 4.3 we can find the same result for $\frac{\partial e[n]}{\partial w}$:

$$\begin{aligned}
\frac{\partial e[n]}{\partial w} &= \sum_{k=0}^{n} \frac{\partial e[n]}{\partial w[n-k]} \\
&= \sum_{k=0}^{n} 2(x[n] - d[n]) w^k x[n-k-1] \\
&= \sum_{k=0}^{n} 2(x[n] - d[n]) \sum_{i=0}^{n-k-1} w^{n-i-1} w_{\text{in}} u[i] \\
&= 2(x[n] - d[n]) w_{\text{in}} \sum_{k=0}^{n} \sum_{i=0}^{n-k-1} w^{n-i-1} u[i] \\
&= 2(x[n] - d[n]) w_{\text{in}} \sum_{i=0}^{n-1} \sum_{k=0}^{n-i-1} w^{n-i-1} u[i] \\
&= 2(x[n] - d[n]) w_{\text{in}} \sum_{i=0}^{n-1} (n-i) w^{n-i-1} u[i]
\end{aligned}$$

In summary, in this toy example, we can see that the norm of the weights (i.e. $|w|$) has a strong influence on (1) the fading memory of the network, and (2) on the vanishing/explosion of the gradient. In order not to be on a chaotic regime we need $|w| \leq 1$. Moreover the closer to 1, the more time it takes for the memory to fade. If $|w| > 1$, this makes explode the gradient to explode and lead to dramatic changes of w in one single learning step. If $|w| < 1$, this makes the gradient vanish, removing thus its dependency on input far back in time. One solution that might solve all problems is to consider $|w| = 1$. In practice, to get something similar to $|w| = 1$, people have tried to use orthogonal recurrent matrices, i.e. matrices through which the norm is conserved (Jing et al., 2017; Jing et al., 2019).

4.4 Reservoir Computing

In order to prevent the gradient from vanishing/exploding the easiest solution is not to train the parameters for which the gradient would vanish/explode. Therefore, the solution proposed by the Reservoir Computing paradigm (RC) is to not train the recurrent weights. The basic idea is that, even though the recurrent weights are not trained, the dynamics of the recurrent layer (a.k.a reservoir) by itself will implicitly act as a temporal filter that will capture relevant aspects of the history of inputs. The question becomes how do we construct reservoirs in order to better represent the input history. Even though there is some advance in the linear case (Tio, 2020), it remains still an open question.

The reason is that there are two conflicting but complementary intuitions that have been followed in the literature. On the first hand, in order for the dynamic to be stable, or robust to disturbances, we need to impose some kind of fading memory on the reservoir. This fading memory property has been defined as the echo state property at the very beginning of reservoir computing (Jaeger, 2001b). A reservoir possesses the echo state property if its dynamic asymptotically depends only on the input signal. The influence of the initial conditions should thus gradually disappear over time. The dynamic would then be an "echo" of the input. On the second hand,

in order to capture relevant aspects of the input, the dynamics should be complex enough. Indeed a very easy way to ensure that the dynamics have the echo state property is to have null recurrent weights. In that case, there is no input history at all in the dynamic and consequently it does not represent any history at all. Here comes the conflict: the more complex the dynamic of the reservoir, the more complex the aspects it can capture, but also the less fading memory it will have. Choosing a good reservoir is thus like making a tradeoff between stability and complexity of the dynamic. One way to solve this is to choose a reservoir whose dynamic is at the edge of chaos (Dambre et al., 2012a). The echo state property seems very important for stability, however in its current form and by definition, reservoir exhibiting the echo state property cannot have several attractors. In order to allow possibly several attractors, the echo state property must be made local. A reservoir would have a local echo state property if, starting from two close reservoir states, its dynamic would asymptotically depend only on the input signal. However, using feedback it is possible to build several attractors (Sussillo and Barak, 2013a; Pascanu and Jaeger, 2011a).

Originally, in the proposal of (Jaeger, 2001b), reservoirs were chosen by randomly sampling their recurrent and input weights and the choice was: "how to scale these weights?". For instance, the stability could be controlled by the spectral radius of the recurrent weights. However, more structured approaches have emerged. For instance, Rodan and Tino proposed a minimalistic approach, where the neurons in the reservoir form a ring structure and there is a single fixed weight between the neurons that are connected in the ring (Rodan and Tino, 2011). Even though the structure of the reservoir is simple and involved no randomness, they have shown, on various tasks, that the performance of these structured reservoirs was similar from the reservoir in the initial proposal of Jaeger. Moreover, in the linear case, this ring structure seems to be theoretically a better solution to design a reservoir (Tio, 2020). In (Voelker, Kajić, and Eliasmith, 2019), authors show how to design a linear reservoir from which we know theoretically how to extract the input history. More recently a layered version of echo state network has been proposed in (Gallicchio, Micheli, and Pedrelli, 2018a).

Even though the reservoir computing approach seems "naive", there is an increasing amount of evidence showing that it can compete with Long Short Term Memory networks (LSTMs, explained in the next section) in synthetic tasks (Jaeger, 2012) as well as in real-world tasks (Gallicchio, Micheli, and Pedrelli, 2018b). Moreover, there is also a universal approximation property for Echo State Network, currently a bit weaker than for classical recurrent neural network (Grigoryeva and Ortega, 2018).

4.4.1 Memory cells (LSTM/GRU)

In order to prevent the gradient from vanishing, Hochreiter and Schmidhuber introduced the idea to use self-loops where the gradients can flow for longer duration. In practice, it is implemented by memory cells in Long Short Term Memory networks (LSTM), whose primary behavior reflects directly these self-loops:

$$\tilde{c}[n] = \tanh(W_{c,c}c[n-1] + W_{c,u}u[n] + b_c)$$
$$c[n] = c[n-1] + \tilde{c}[n]$$

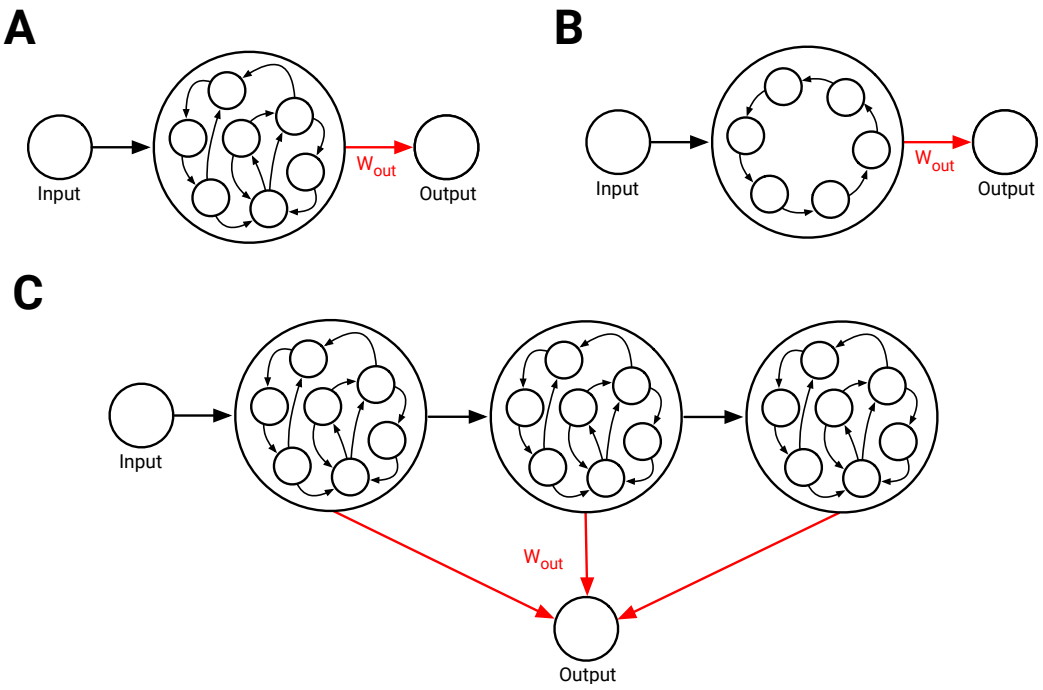


FIGURE 4.4: Reservoir approach. Only output weights (red) are trained. **A**. Random initialization of the recurrent weights (Jaeger, 2001b). **B**. Deterministic ring structured initialization of the recurrent weights (Rodan and Tino, 2011). **C**. Random layered initialization of the recurrent weights (Gallicchio, Micheli, and Pedrelli, 2018b).

where c is the state of the cell, $\tilde{c}$ is what is added to the state of the cell, and $W_{c,c}$, $W_{c,u}$ and b_c are weights.

In addition to this, they proposed to use gates to monitor the state of the cell. In a way, the memory cells are like working memories controlled by these gates. First, they use an input gate which will control when information should enter the cell:

$$c[n] = c[n-1] + i[n]\tilde{c}[n]$$

where i, the state of the input gate, is a signal between 0 and 1 controlling how much should be added to the state of the cell. Second, they use an output gate which controls when information in the cell should be spread:

$$c[n] = c[n-1] + \tilde{c}[n]$$
$$h[n] = o[n]\tanh(c[n])$$

where the state of the cells c is decoupled from their output h, and o is the state of the output gate which is a signal between 0 and 1 controlling how much the state of the cell shoud be spread to other cells.

Later on, a forget gate was introduced (Gers and Schmidhuber, 2000), which allow to control the strength of these self-loops depending on the context.

$$c[n] = f[n]c[n-1] + \tilde{c}[n]$$

where f, the state of the forget gate, is a signal between 0 and 1 controlling the strength of the self-loop.

The overall dynamics of LSTM that combines all the gates is described by the following set of equations:

$$
\begin{aligned}
i[n] &= \sigma(W_{i,h}h[n-1] + W_{i,u}u[n] + b_i) \\
f[n] &= \sigma(W_{f,h}h[n-1] + W_{i,u}u[n] + b_f) \\
o[n] &= \sigma(W_{o,h}h[n-1] + W_{i,x}u[n] + b_o) \\
\tilde{c}[n] &= \tanh(W_{c,c}c[n-1] + W_{c,u}u[n] + b_c)c[n] \qquad = f[n]c[n-1] + i[n]\tilde{c}[n] \\
h[n] &= o[n]\tanh(c[n])
\end{aligned}
$$

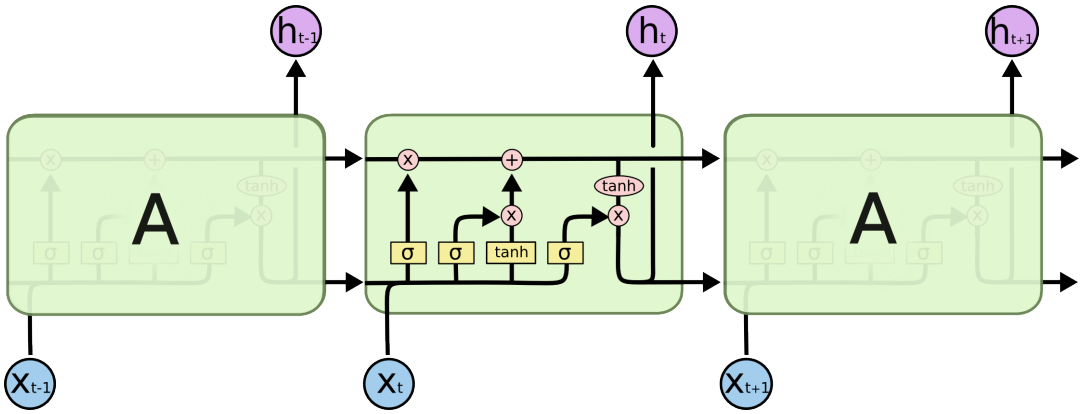


FIGURE 4.5: Long Short Term Mermory network (LSTM) architecture.
Figure extracted from (Christopher Olah, 2015)

There have been many variants of memory cells, for a recent review on these variants and how they are used in a network see (Yu et al., 2019). For instance peephole connections have been introduced in order to allow the gates to inspect the current internal states of the cells (Gers and Schmidhuber, 2000). The input gate dynamic would be re-written as:

$$i[n] = \sigma(W_{i,c}c[n-1] + W_{i,h}h[n-1] + W_{i,u}u[n] + b_i)$$

However, this adds a lot of extra parameters to train. Other variants have been introduced in order to reduce the number of parameters. For instance, if we rely on the fact that, most likely, when we want to store new things, we also want to forget others, it is possible to remove parameters by coupling the input gate and the forget gate. That is precisely how Gated Recurrent Unit (GRU) have been proposed (Cho et al., 2014b). In GRU, the update gate combines both the input and forget gate

$$c[n] = (1 - z[n])c[n-1] + z[n]\tilde{c}[n]$$

where z, the state of the update gate, is a signal between 0 and 1 controlling how much the state of the cell should be updated. In GRU, the state of the memory cell and its output are merged, and a reset gate simplifies the output gate.

$$\tilde{c}[n] = \tanh(W_{c,c}r[n]c[n-1] + W_{c,u}u[n] + b_c)$$

Applying the same intuitive idea as before, minimal gated unit reduce even further the number of parameters and combine both update gate and reset gate, and use

a single gate for both (Zhou et al., 2015). So far none of these variants seems to improve significantly the performances of vanilla LSTM (Greff et al., 2017a).

4.5 Explicit memory mechanisms (Memory networks/NTM)

The memory cells in LSTMs can be largely considered as working memories that the gates learn to use. In that regards, Weston, Chopra, and Bordes implement a working memory for the RNN (Weston, Chopra, and Bordes, 2014). More specifically, they proposed the memory networks in which they make an RNN interact with an external memory. This external memory can be used via an addressing mechanism. Initially, this addressing mechanism required supervision in order to be used. However, around the same time, (Graves, Wayne, and Danihelka, 2014) proposed the Neural Turing Machines in which this addressing doesn't require pre-training. With a similar content addressing mechanism (Sukhbaatar et al., 2015) removed the supervision from the memory networks.

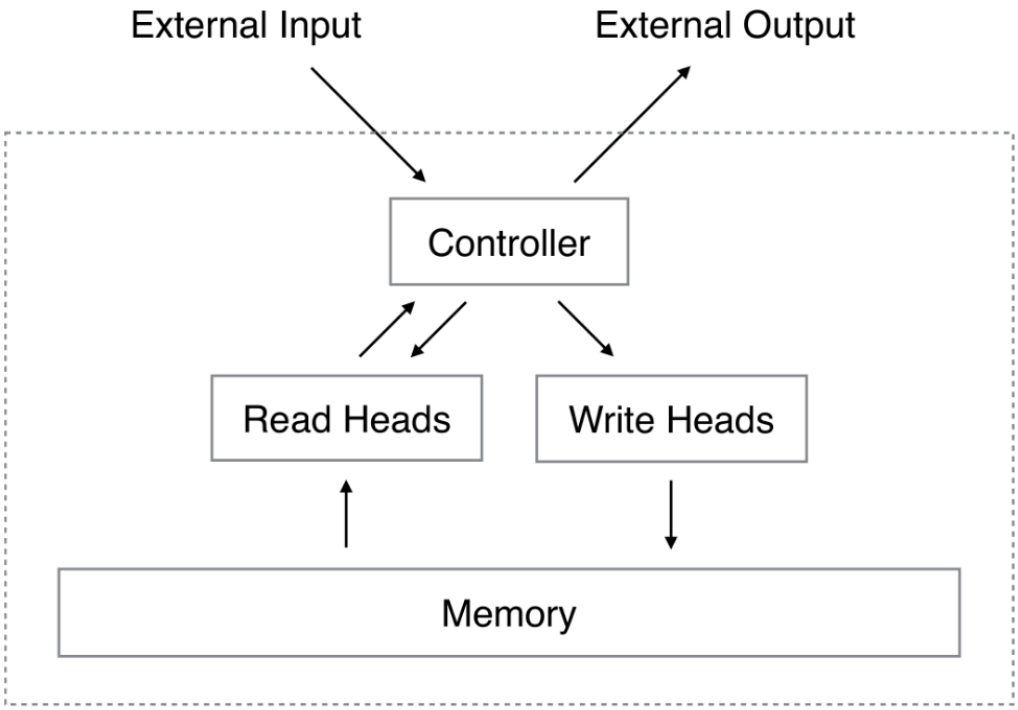


FIGURE 4.6: Neural Turing Machine (NTM) architecture. Figure extracted from (Graves, Wayne, and Danihelka, 2014)

Let us describe in more details the neural turing machines (NTMs). With NTM, Graves, Wayne, and Danihelka attempted to create an analog of a Turing Machine in an RNN (Graves, Wayne, and Danihelka, 2014). A Turing machine is composed of three components: (1) a tape of infinite length where symbols are written, (2) a tape head placed on the tape which can move along the tape, read the content of the tape, or write new content on the tape, and (3) a finite state machine which determines what action to do with the head (move/read/write). As shown in Figure 4.6, each of these components have their analog in the NTM: (1) a tape of finite length where, instead of symbols, vectors of real numbers are written, (2) a read/write head that

no longer has a physical position on the tape but will be able to read/write memories through an addressing mechanism, and (3) an RNN controller that not only has access to information read by the head but also to external input, and that decides where to read/write on the tape. In a Turing machine, all operations are performed on the tape: the input is the initial state of the tape and the output is the final state. Whereas in NTMs, the inputs/outputs are separated from the tape, the RNN controller receives the inputs and produces the outputs. The head of an NTM combines two addressing mechanism: a content-based mechanism and a location-based mechanism. To have differentiable addressing mechanisms, both rely on the same idea: the head will read and write everywhere, but with different levels of intensity. At each time step, each location in memory is thus associated with a weight. Reading consists in giving as input to the RNN controller the weighted sum of the memories, whereas writing is decomposed in two steps: (1) an erasing step, and (2) an adding step. For writing, these weights allow focusing on where to erase and where to add. The content-based mechanism attempts to select a memory that is the best aligned with a key produced by the controller, whereas the location-based mechanism attempts to select a memory given its location. The analogy that NTM has started to make with computers has been extended with Neural Programmer (Neelakantan, Le, and Sutskever, 2015). In computers we have more than reading and writing in memory, there are processing units able to do logic/arithmetic. With Neural Programmer, Neelakantan, Le, and Sutskever proposed to incorporate such processing units in an RNN.

4.6 Summary

In this chapter, a direct analogy has been made between working memory and long-term time dependency. We have also seen why, in a simplified example of SRN, learning long-term time dependencies is hard in practice for Simple Recurrent Networks (SRNs). Training an SRN to solve working memory tasks is therefore complex. However, solutions have been proposed. It is interesting to note that two of them actually involve incorporating working memory into the model. In the first one, Long Short-Term Memory networks (LSTMs), the working memory mechanism (i.e. gating) is added directly to the units of the network. In the second one, an external memory is added with handcrafted read/write operations. The solution proposed by the Reservoir Computing paradigm (RC) is the least hand-crafted because no explicit mechanism directly related to working memory is added. In this thesis, we propose to explore how such RC models can still be used to obtain a generic working memory mechanism.

Chapter 5

A generic working memory mechanism: Gating

As shown in Chapter 3, a lot of models of working memory are handcrafted, i.e. how the model will maintain is cleverly designed. In this chapter we propose another way to build a model for working memory. We train a recurrent neural network (RNN) to perform a generic working memory task: Gating. What can be viewed as handcrafted in our model is the function that will be performed by the model, i.e. the gating task, and not how the model will implement the function, i.e. how the model will maintain via gating.

In fact, back in the early 90s Zipser et al. have already shown how it is possible to perform such gating using a recurrent neural network which learn through backpropagation through time Zipser et al., 1993, and how the firing rate activity of such model relates to neurons recorded in the prefrontal cortex (PFC) in many different experiments. If they show that gating is a plausible candidate for a generic mechanism of working memory, they do not explain how the recurrent neural network obtained is actually performing the gating.

In this chapter we show: (1) how such gating can be learned by a way simpler class of recurrent neural networks, reservoirs, and (2) how such reservoir implement gating. As shown in Chapter 4, reservoirs are simpler because their training do not require the use any time unfolding, i.e. there is no need for any backpropagation through time. In fact, reservoirs are known to have dynamics similar to those of the prefrontal cortex (PFC), a brain region essential for working memory (Enel et al., 2016a). Moreover, despite their simplicity, reservoirs have been shown able to handle long-term time dependencies, which is a key to working memory (Jaeger, 2012).

NEURAL COMPUTATION

JANUARY 2020

A Robust Model of Gated Working Memory

Abstract: Gated working memory is defined as the capacity of holding arbitrary information at any time in order to be used at a later time. Based on electrophysiological recordings, several computational models have tackled the problem using dedicated and explicit mechanisms. We propose instead to consider an implicit mechanism based on a random recurrent neural network. We introduce a robust yet simple reservoir model of gated working memory with instantaneous updates. The model is able to store an arbitrary real value at random time over an extended period of time. The dynamics of the model is a line attractor that learns to exploit reentry and a non-linearity during the training phase using only a few representative values. A deeper study of the model shows that there is actually a large range of hyper parameters for which the results hold (number of neurons, sparsity, global weight scaling, etc.) such that any large enough population, mixing excitatory and inhibitory neurons can quickly learn to realize such gated working memory. In a nutshell, with a minimal set of hypotheses, we show that we can have a robust model of working memory. This suggests this property could be an implicit property of any random population, that can be acquired through learning. Furthermore, considering working memory to be a physically open but functionally closed system, we give account on some counter-intuitive electrophysiological recordings.

Keywords: Working Memory, Line Attractor, Reservoir Computing, Computational Neuroscience, Prefrontal Cortex, Model

Re-editing from the published paper (Strock, Hinaut, and Rougier, 2020)

5.1 Introduction

The prefrontal cortex (PFC), noteworthy for its highly recurrent connections (GoldmanRakic, 1987), is involved in many high level capabilities, such as decision making (Bechara et al., 1998), working memory (GoldmanRakic, 1987), goal directed behavior (Miller and Cohen, 2001), temporal organisation and reasoning (Fuster, 2001). In this article, we are more specifically interested in gated working memory (O'Reilly and Frank, 2006b) that is defined as the capacity of holding arbitrary information at a given random time t_0 such as to be accessible at a later random time t_1 (see Figure 5.1). Between times t_0 and t_1, we make no assumption on the inner mechanisms of the working memory. The only measures we are interested in are the precision of the output (compared to the initial information) and the maximal delay during which this information can be accessed within a given precision range. One obvious and immediate solution to the task is to make an explicit copy (inside the memory) of the information at time t_0 and to hold it unchanged until it is read at time t_1, much like a computer program variable that is first assigned a value in order to be read later. Such solution can be easily characterized by a fixed pattern of sustained activities inside the memory. This is precisely what led researchers to search for such sustained activity inside the frontal cortex (Funahashi, 2017b; Constantinidis et al., 2018b), where an important part of our working memory capacities is believed to be located. Romo et al. (1999) have shown that PFC neurons of non-human primates can maintain information about a stimulus for several seconds. Their firing rate was correlated with the coding of a specific dimension (frequency) of the stimulus maintained in memory. However, when Machens, Romo, and Brody (2010) later re-analyzed the data of this experiment, they showed that the stimulus was actually encoded over a sub-population using a distributed representation. Similarly, when Rigotti et al. (2013b) analyzed single neuron activity recorded in the lateral PFC of monkeys performing complex cognitive tasks, they found several neurons displaying task-related activity. Once they discarded all the neurons that were displaying task-related activity, they were still able to decode task information with a linear decoder. They proposed that the PFC hosts high-dimensional linear and non-linear mixed-selectivity activity. The question is thus, if working memory is not encoded in the sustained activity, what can be the alternatives?

Before answering that question, let us first characterize the type and the properties of information we consider before defining what it means to access the information.

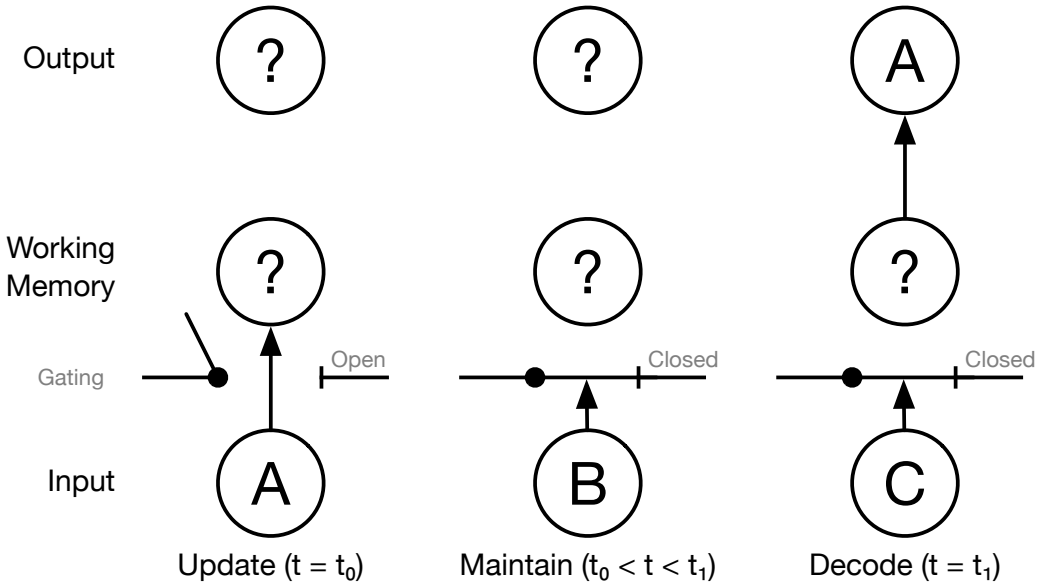


FIGURE 5.1: **Gated working memory** is defined as the capacity of holding arbitrary information at a given random time t_0 such as to be accessible at a later random time t_1. Between times t_0 and t_1, we make no assumption on the activity inside the working memory. Note that a closed gate does not mean external activities cannot enter the memory.

The type of information that can be stored inside working memory has been characterized using different cognitive tasks such as for example the delayed matching-to-sample (DMTS), the N-back task or the Wisconsin Card Sorting Task (WCST). From these different tasks, we can assume that virtually any kind of information, be it an auditory or visual stimulus, textual or verbal instruction, implicit or explicit cue, is susceptible to be memorized and processed inside the working memory. From a computational point of view, this can be abstracted into a set of categorical, discrete or continuous values. In this work, we are only interested in the most general case, the continuous one, which can be reduced into a single scalar value. The question we want to address is how a neural population can gate and maintain an arbitrary (e.g. random) value, in spite of noise and distractors, such that it can be decoded non-ambiguously at a later time.

To answer this question we can search the extensive literature on computational models of working memory that have been extensively reviewed by Durstewitz, Seamans, and Sejnowski (2000), by Compte (2006) and more recently by Barak and Tsodyks (2014). More specifically, Compte (2006) and Durstewitz, Seamans, and Sejnowski (2000) explains that the retention of information is often associated with attractors. In the simplest case, a continuous scalar information can be identified with a position on a line attractor. However, this kind of memory exhibits stability issues: unlike a point attractor (i.e. a stable fixed point), a line attractor is marginally stable such that it cannot be robust against all small perturbations. Such a line attractor can be stable against orthogonal perturbations, but not against colinear perturbations (i.e. perturbations along the line). Furthermore, the design (and

numerical implementation) of a line attractor is tricky because even small imperfections (e.g. numerical errors) can lead to instability. Nevertheless, there exist several models that can overcome these limitations.

This is the case for the theoretical model by Amari (1977) who proved that a local excitation could persist in the absence of stimuli, in the form of a localized bump of activity in an homogeneous and isotropic neural field model, using long range inhibition and short range excitation. This model represents *de facto* a spatial attractor formed by a collection of localized bumps of activity over the plane. A few decades later, Compte (2000) showed that the same lasting bump property can also be achieved using leaky integrate and fire neurons arranged on a ring, with short range excitation and constant inhibition (ring bump attractor). This model has since then been extended (Edin et al., 2009; Wei, Wang, and Wang, 2012) with the characterization of the conditions allowing to have simultaneous bumps of activity. This would explain multi-item memorization where each bump represents a different information that is maintained simultaneously with the other bumps. Similarly, Bouchacourt and Buschman (2019a) proposed to handle multi-items memorization by duplicating the bump attractor model. They explicitly limited the number of items to be maintained in memory through the interaction between the different bump attractor models (using a random layer of neurons). If all these models can cope with the memorization of a graded information, this information is precisely localized in the bumps of activity and corresponds to a sustained activity. Such patterns of activity have been identified in several cerebral structures (e.g. head direction cells (Zhang, 1996) in mammals, superior colliculus (Gandhi and Katnani, 2011) in primates) but it is not yet clear to what extent this can give account of a general working memory mechanism. Such sustained activity is also present in the model of Koulakov et al. (2002) who consider a population of bistable units that encodes a (quasi) graded information using distributed encoding (percentage of units in high state). This solves both the robustness and stability issue of the line attractor by virtue of discretization. Finally, some authors (Zipser et al., 1993; Lim and Goldman, 2013a) consider the encoding of the value to be correlated with the firing rate of a neuron or of a group of neurons. This is the case for the model proposed by Lim and Goldman (2013a) who obtain stability of the firing rate by adding a negative derivative self feedback (hence artificially increasing the time constant of neurons). They show how such mechanism can be implemented by the interaction of two populations of excitatory and inhibitory neurons evolving at different time scales. However, independently of the encoding of the graded value, most of model authors are interested in characterizing the mechanism responsible for the maintenance property. They tend to consider the memory as an isolated system, not prone to external perturbations, with the noticeable exception of the model by Zipser et al. (1993) which is constantly fed by an input.

In this work, we consider working memory to be an open system under the constant influence of external activities (i.e. even when the gate is *closed*). Thus, we can not rely on a dynamical system that hosts a line attractor in

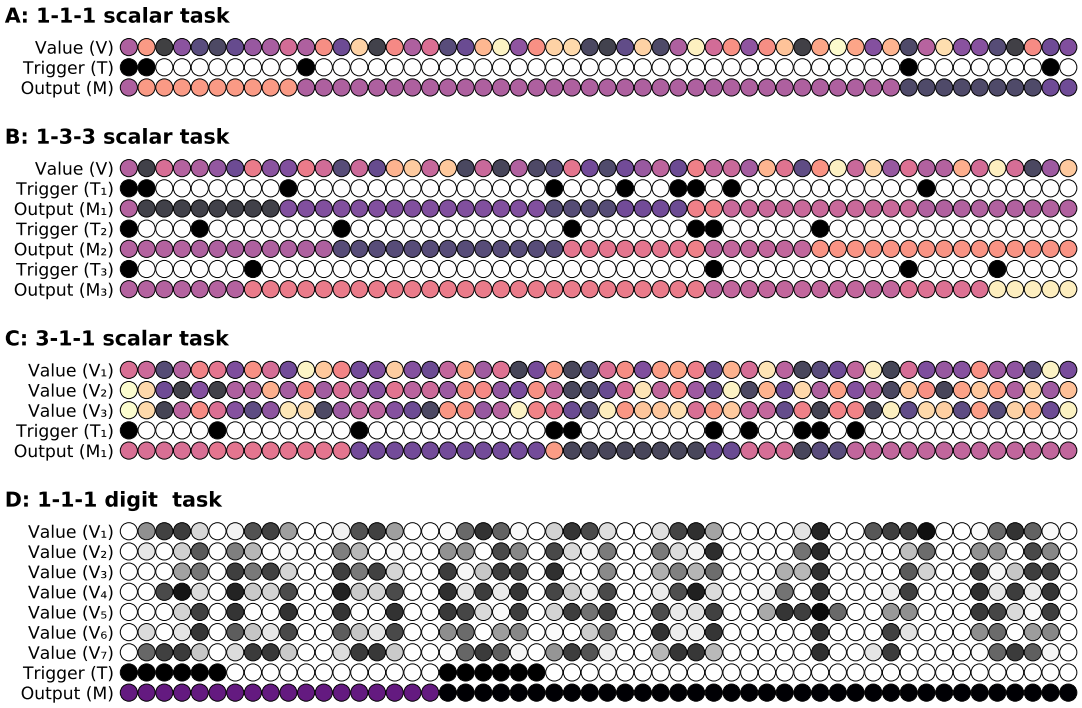


FIGURE 5.2: **Working memory tasks** Each column represents a time step (time increases from left to right), colored discs represent activity in the input (V or V_i and T or T_i) and the output (M or M_i). **A.** Gated working memory task with one gate. **B.** Gated working memory task with three gates. **C.** Gated working memory task with one gate but three inputs. To solve the task, it is necessary to ignore the two irrelevant inputs. **D.** Gated working memory task with one gate where the scalar input V has been replaced by a streamed input of bitmap digits. To solve the task, it is thus necessary to recognize the digits and to transform them into a normalized value.

the absence of inputs. We have to design an input-dependent dynamical system that is robust against all kinds of perturbations (input, internal, output feedback). First, we will formalize a set of tasks that will be used to study features and performances of the different models we have considered. Then, we will introduce a minimal model that will help us in explaining the mechanism needed for a more general model. For this general one, we will consider a particular instance of reservoir: namely an Echo State Network (ESN) (Jaeger, 2001a). The analysis of this model will allow us to show that reservoir activity is characterized by a combination of both sustained and transient activities. Moreover, we will show that none of these activities are critical for the decoding of the correct output. Finally, we will show that in the absence of input the dynamics of the model implements a segment attractor (i.e. a line attractor with bounding values).

5.2 Methods

In this section, we formalize and extend the gated working memory task that has been described in the introduction (see Figure 5.1). We consider

four different tasks that will illustrate the generic capacity of the reservoir model to store continuous values or a discrete set of values. The first three are variations of the WM task for continuous values with various number of input values (n-value) and gating WM units (n-gate). The last task includes a non-linear computation (i.e. digit recognition from pixels) in addition to the gating task for discrete values.

5.2.1 The n-value p-gate scalar working memory tasks

In the 1-value 1-gate scalar version of the task, we consider at time $t \in \mathbb{N}$ an input signal $V(t)$ in [-1,+1], an input trigger $T(t)$ in {0, 1} and an output $M(t)$ that is defined as the value of $V(t^*)$ where t^* is the most recent time such that $T(t^*) = 1$ (see Figure 5.2A). This can be written as:

$$M(t) = V(t^*) \text{ with } t^* = \max_{0 \leq t' \leq t} (t' \mid T(t') = 1) \tag{5.1}$$

Said differently, the output is the value of the input when the gate was open for the last time. Note that this can be also rewritten as a simple select operator between the input V and the output M:

$$M(t) = T(t)V(t) + (1 - T(t))M(t-1) \tag{5.2}$$

In the 1-value 3-gate version of the task, we consider at time $t \in \mathbb{N}$ an input signal $V(t)$ in [-1,+1], three input triggers $T_{\{1,2,3\}}(t)$ in {0, 1} and three outputs $M_{\{1,2,3\}}(t)$ that are respectively defined as the value of $V(t^*)$ where t^* is the most recent time such that $T_{\{1,2,3\}}(t^*) = 1$ (see Figure 5.2B). This can be written as:

$$M_i(t) = M_i(t^*) \text{ with } t^* = \max_{0 \leq t' \leq t} (t' \mid T_i(t') = 1), i \in \{1, 2, 3\} \tag{5.3}$$

In the 3-value 1-gate scalar version of the task, we consider at time $t \in \mathbb{N}$ three input signals $V_{\{1,2,3\}}(t)$ in [-1,+1], an input trigger $T(t)$ in {0, 1} and an output $M(t)$ that is defined as the value of $V_1(t^*)$ where t^* is the most recent time such that $T(t^*) = 1$ (see Figure 5.2C). This can be written as:

$$M(t) = V_1(t^*) \text{ with } t^* = \max_{0 \leq t' \leq t} (t' \mid T(t') = 1) \tag{5.4}$$

This can be easily generalized to a *n-value p-gate scalar task* with n input signals, p input triggers, and p outputs. Only the first input signal and the t input triggers determines the p outputs. The other $n - 1$ input signals are additional inputs irrelevant to the task.

5.2.2 The digit 1-value 1-gate working memory task

In the digit version of the 1-value 1-gate working memory task, we define a slightly more complex version of the V input by considering a bitmap representation of it (see Figure 5.2D). Using a monotype font (Inconsolata Regular[1]) at size 11 points, we draw a grayscaled bitmap representation of a sequence of random digits (0 to 9), each digit being of size 6×7 pixels (after having cropped top and bottom empty lines) and the trigger signal being expanded to the width of a glyph. The output is defined as a discrete and normalized value. It is to be noted that there is no possible linear interpolation between the different inputs, as it was the case for the scalar version. Formally, we can define the output as:

$$M(t) = f(V_{i\in[1,7]}(t^*), V_{i\in[1,7]}(t^*-1), ..., V_{i\in[1,7]}(t^*-5)) \text{ with } t^* = \max_{0\leq t'\leq t}(t' \mid T(t') = 1) \tag{5.5}$$

with f a function that maps the representation of a digit to a normalized value. Since there are 10 values, the digit n is associated to $\frac{n}{10}$. This function has to be learned by the model in order to solve the task. It would have been possible to use the MNIST database (Schaetti, Salomon, and Couturier, 2016) instead of a regular font but this would have also complexified the task, and make the training period much longer, because a digit is processed only when a trigger is present. If we consider for example a sequence of 25,000 digits and a trigger probability of 0.01, this represents (in average) 250 triggers for the whole sequence and consequently only 25 presentations per digit. In comparison, MNIST train data set has 60,000 digits, which would have required as many triggers, and a hundred times more digits.

5.2.3 The minimal model

It is possible to define a minimal degenerated reservoir model (if we consider X_1, X_2 and X_3 to be the reservoir) that takes explicitly advantage of the non-linearity to perform the gating mechanism, as shown in Figure 5.3. There is no learning in this model. It is parametrized by two values a (large enough) and b (small enough) and it is fully defined by the following set of equations:

$$\begin{aligned} X_1[n+1] &= \tanh(b\ V[n]) \\ X_2[n+1] &= \tanh(b\ V[n] + a\ T[n]) \\ X_3[n+1] &= \tanh(b\ M[n] + a\ T[n]) \\ M[n+1] &= (X_1[n+1] - X_2[n+1] + X_3[n+1])/b \end{aligned} \tag{5.6}$$

[1]The Inconsolata font is available from https://www.levien.com/type/myfonts/inconsolata.html

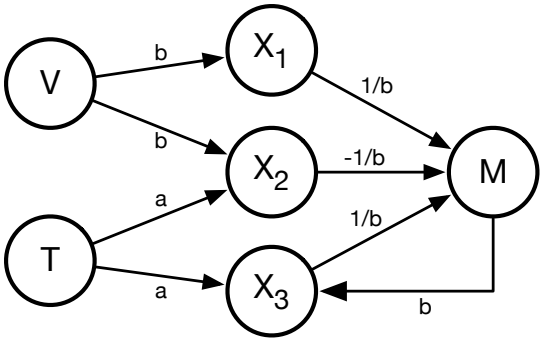


FIGURE 5.3: **A minimal gated working memory model** is able to solve the 1-value 1-gate working memory task by taking advantage of the non-linearity and asymptotic behavior of tanh units. Performance is controlled with parameters a and b. This architecture can be generalized to the *n-value p-gate task* by adding more than one reservoir unit (corresponding to X_3) for each supplementary trigger/output couple.

In order to understand how this minimal model works, we can write the output $M[n]$ relatively to the value of the trigger $T[n]$ which lies in $\{0, 1\}$:

$$\text{If } T[n] = 0 \begin{cases} X_1[n+1] &= \tanh(b\ V[n]) \\ X_2[n+1] &= \tanh(b\ V[n]) \\ X_3[n+1] &= \tanh(b\ M[n]) \\ M[n+1] &= \tanh(b\ M[n])/b \end{cases} \tag{5.7}$$

When $T[n] = 0$, if we assign b a small value (e.g. $b = 10^{-3}$) and considering that $\tanh(x) \underset{x \to 0}{\sim} x$, from equation (5.7), we have $M[n+1] \approx M[n]$.

$$\text{If } T[n] = 1 \begin{cases} X_1[n+1] &= \tanh(b\ V[n]) \\ X_2[n+1] &= \tanh(b\ V[n] + a) \\ X_3[n+1] &= \tanh(b\ M[n] + a) \\ M[n+1] &= \tanh(b\ V[n])/b \\ & \quad -(\tanh(b\ V[n] + a) - \tanh(b\ M[n] + a))/b \end{cases} \tag{5.8}$$

When $T[n] = 1$, if we assign a to a large value (e.g. $a = 1000$) and considering that b is small and that $\lim_{x \to \infty} \tanh(x) = 1$, we have $\tanh(b\ V[n] + a) \approx \tanh(b\ M[n] + a) \approx 1$. From equation (5.8), we then have $M[n+1] \approx \tanh(b\ V[n])/b \approx V[n]$.

Consequently, the trigger $T[n]$ fully dictates the output. When $T[n] = 0$, the output $M[n+1]$ is unchanged and corresponds to the current memory ($M[n]$). When $T[n] = 1$, the output $M[n+1]$ is assigned the value $V[n]$ of the input. We think this model represents the minimal model that is able to solve the gating working memory task using such simple neurons (with tanh activation function). By taking advantage of the linear regime around 0 and the asymptotic behavior around infinity, this model gracefully solves the task using only two parameters (a and b). However, we have no formal proof that

a model having only two neurons in the reservoir part cannot solve the task.

Note that this minimal model turns out to be similar to the memory cell of a Gated Recurrent Unit (Cho et al., 2014a) (GRU) without its reset gate[2], but using only simple tanh neurons, in comparison to hand-crafted LSTM-like cells. Without the reset gate, the dynamics of a GRU cell can be simplified to:

$$h[n+1] = (1 - z[n+1])h[n] + z[n+1]x[n+1] \tag{5.9}$$

where h is the state of the memory cell, z is the update gate neuron, and x is an input neuron. By using functional equations of hyperbolic functions and classical order 1 Taylor expansions, and for the sake of simplicity by replacing all the $o_{b\to 0}(b)$ by $o(b)$, we can obtain a similar equation in the minimal model:

$$\begin{aligned}
M[n+1] &= \frac{\tanh(b\,M[n] + a\,T[n]) - \tanh(b\,V[n] + a\,T[n]) + \tanh(b\,V[n])}{b} \\
&= \frac{\tanh(b\,M[n] + a\,T[n]) - \tanh(b\,V[n] + a\,T[n])}{b} + \frac{\tanh(b\,V[n])}{b} \\
&= \frac{\sinh(b\,(M[n] - V[n]))}{b(\cosh(b\,M[n] + a\,T[n])\cosh(b\,V[n] + a\,T[n]))} + \frac{\tanh(b\,V[n])}{b} \\
&= \frac{\sinh(b\,(M[n] - V[n]))}{b}\frac{1}{\cosh(b\,M[n] + a\,T[n])\cosh(b\,V[n] + a\,T[n])} + \frac{\tanh(b\,V[n]}{b} \\
&= \frac{b\,(M[n] - V[n]) + o(b)}{b}\left(\frac{1}{\cosh(a\,T[n])^2} + o(1)\right) + \frac{b\,V[n] + o(b)}{b} \\
&= \frac{(M[n] - V[n])}{\cosh(a\,T[n])^2} + V[n] + o(1) \\
&= (M[n] - V[n])(1 - \tanh(a\,T[n])^2) + V[n] + o(1) \\
&= (1 - \tanh(a\,T[n])^2)M[n] + \tanh(a\,T[n])^2 V[n] + o(1)
\end{aligned}$$

If b is small enough, we have $M[n+1] \approx (1 - \tanh(a\,T[n])^2)M[n] + \tanh(a\,T[n])^2 V[n]$ and consequently, T acts as an update gate for M. The final form that is equivalent to the GRU without reset gate (as in equation 5.9) is given by:

$$M[n+1] = (1 - z[n+1])M[n] + z[n+1]V[n+1] \tag{5.10}$$

with $z[n] \approx (1 - \tanh(a\,T[n])^2)$.

5.2.4 The reservoir model

We consider an echo state network (Jaeger, 2001a) (see Figure 5.4) with leaky neurons, and feedback from readout units to the reservoir, which is described

[2] We can actually find a similar LSTM variant in Greff et al., 2017b named *Coupled Input and Forget Gate*

by the following update equations:

$$\begin{aligned} x[n] &= (1-\alpha)x[n-1] \\ &\quad + \alpha \tanh\left(W_{in}u[n] + W(x[n-1] + \xi) + W_{fb}(y[n-1])\right) \\ y[n] &= W_{out}x[n] \end{aligned} \tag{5.11}$$

where $u[n]$, $x[n]$ and $y[n]$ are respectively the input, the reservoir and the output at time n. W, W_{in}, W_{fb} and W_{out} are respectively the recurrent, the input, the feedback and the output weight matrix and α is the leaking rate. ξ is a uniform white noise term added to the reservoir: ξ is independent for each neuron. During initialization, matrices W_{in} and W_{fb} are sampled

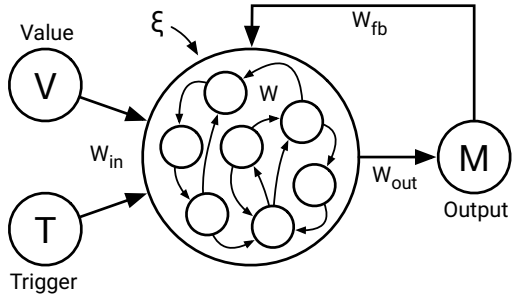


FIGURE 5.4: **The reservoir model** receives a random signal V in $[-1,+1]$ and a trigger signal T in $\{0,1\}$. The reservoir is made of non-linear units (tanh) and the output M is fed back to the reservoir at each iteration. For *n-gate p-value task*, we use n triggers, p values and n outputs. For a generic approach, we use notation u for the input, x for the reservoir and y for the output, independently of the task.

randomly and uniformly between -1 and $+1$ and multiplied by the *input scaling factor* and the *feedback scaling factor* respectively. Matrix W is sampled randomly between -1 and $+1$ and we ensure the matrix enforces the defined *sparsity* (ratio of non null values). The resulting matrix is then divided by its largest absolute eigenvalue and multiplied by the specified *spectral radius*. The matrix W_{out} is initially undefined and is learned through teacher forcing (Lukoeviius, 2012) and a linear regression:

$$W_{out} = YX^T(XX^T)^{-1} \tag{5.12}$$

where Y and X corresponds to the respective concatenation of all the desired outputs and reservoir states during a run, each row corresponding to a time step.

5.3 Results

5.3.1 The reduced model

The reduced model displays a quasi-perfect performance (RMSE = 2e-6 with $a = 10$ and $b = 10^{-3}$) as shown in Figure 5.5 and the three neurons $X1$, $X2$ and $X3$ behave as expected. $X1$ is strongly correlated with V (Figure 5.5B), $X2$ is strongly correlated with V and saturates in the presence of a tick in T (Figure 5.5C) and $X3$ is strongly correlated with M and saturates in the presence of a tick in T (Figure 5.5D). This reduced model is actually a very good approximation of a line attractor (i.e. a line of points with very slow dynamics) even though we can prove that, due to the tanh non-linearity, in the absence of inputs, the model will converge to a null state (possibly after a very long time), independently of parameters a and b and the initial state. Nonetheless, Figure 5.5 clearly shows that information can be maintained provided b is small enough. There is a drift, but this drift is so small that it can be considered negligible relative to the system time constants: these slow points can be considered as a *line* or *segment attractor* (Seung, 1996; Sussillo and Barak, 2013b). As explained by Seung (1998), *the reader should be cautioned that the term "continuous attractor" is an idealization and should not be taken too literally. In real networks, a continuous attractor is only approximated by a manifold in state space along which drift is very slow*. Nevertheless, it is worth mentioning that in order to have a true line attractor, one can replace the tanh activity function with a linear function saturated to 1 and -1 (Seung, 1996).

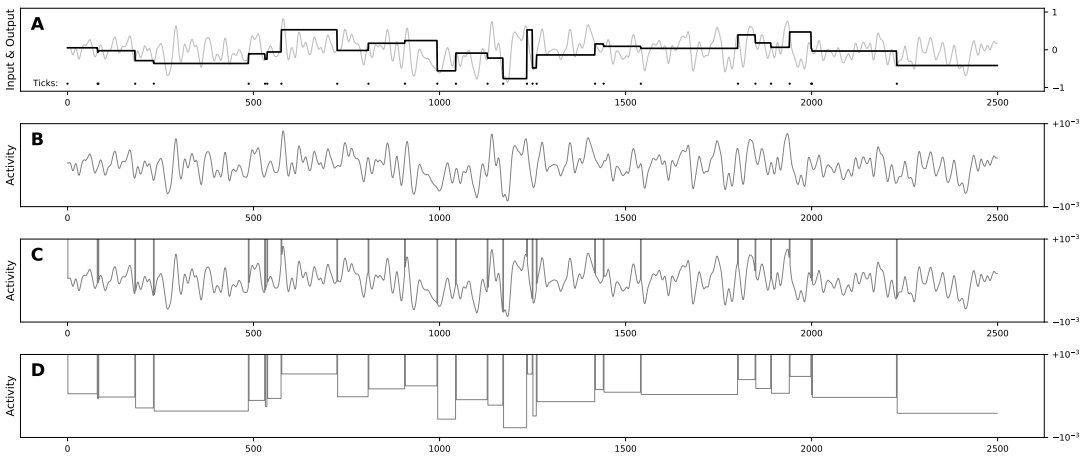


FIGURE 5.5: **Performance of the reduced model on the 1-gate task.** **A** The light gray line is the input signal and the thick black one is the output of the model. Black dots at the bottom represents the trigger signal (when to memorize a new value). **B, C, D** Respective activity of X_1, X_2 and X_3 units.

5.3.2 The reservoir model

Unless specified otherwise, all the reservoir models were parameterized using values given in table 5.1. These values were chosen to be simple and

do not really impact the performance of the model as it will be explained in the Robustness section. All simulations and figures were produced using the

Parameter	Value
Spectral radius	0.1
Sparsity	0.5
Leak	1.0 (no leak)
Input scaling	1.0
Feedback scaling	1.0
Number of units	1000
Noise	0.0001
Training timesteps	25,000
Testing timesteps	2,500
Trigger probability	0.01

TABLE 5.1: **Default parameters** Unless specified otherwise, these are the parameters used in all the simulations.

Python scientific stack, namely, SciPy (Jones, Oliphant, and Peterson, 2001), Matplotlib (Hunter, 2007) and NumPy (Walt, Colbert, and Varoquaux, 2011). Sources are available at github.com/rougier/ESN-WM.

Results for the reservoir model show a very good generalization performance with a precision of the order of 10^{-3} for the level of noise considered (10^{-4}). Better precision can be obtained for lower noise levels, as show in Figure 5.7. Surprisingly, this generalization property stands with as few as only four random training values where we can achieve a 10^{-3} level of precision.

1-value 1-gate scalar task

The model has been trained using parameters given in table 5.1. The V signal is made of 25,000 random values sampled from a pseudo-random uniform distribution between -1 and +1. The T signal is built from 25,000 random binary values with probability 0.01 of having $T = 1$, and $T = 0$ otherwise. During training, each of the input is presented to the model and the output is forced with the last triggered input. All the input (u) and internal (x) states are collected and the matrix W_{out} is computed according to equation (5.12). The model has been then tested using a V signal made of 2500 random values sampled from a pseudo-random uniform distribution between -1 and +1. For readability of the figure (see Figure 5.6A), this signal has been smoothed using a fixed-size Hann window filter. The corresponding T signal has been generated following the same procedure as during the training stage. Figure 5.6A displays an illustrative test run of the model (for a more thorough analysis of the performance, see the Robustness section) where the error in the output is always kept below 10^{-2} and the RMSE is about $3 * 10^{-3}$.

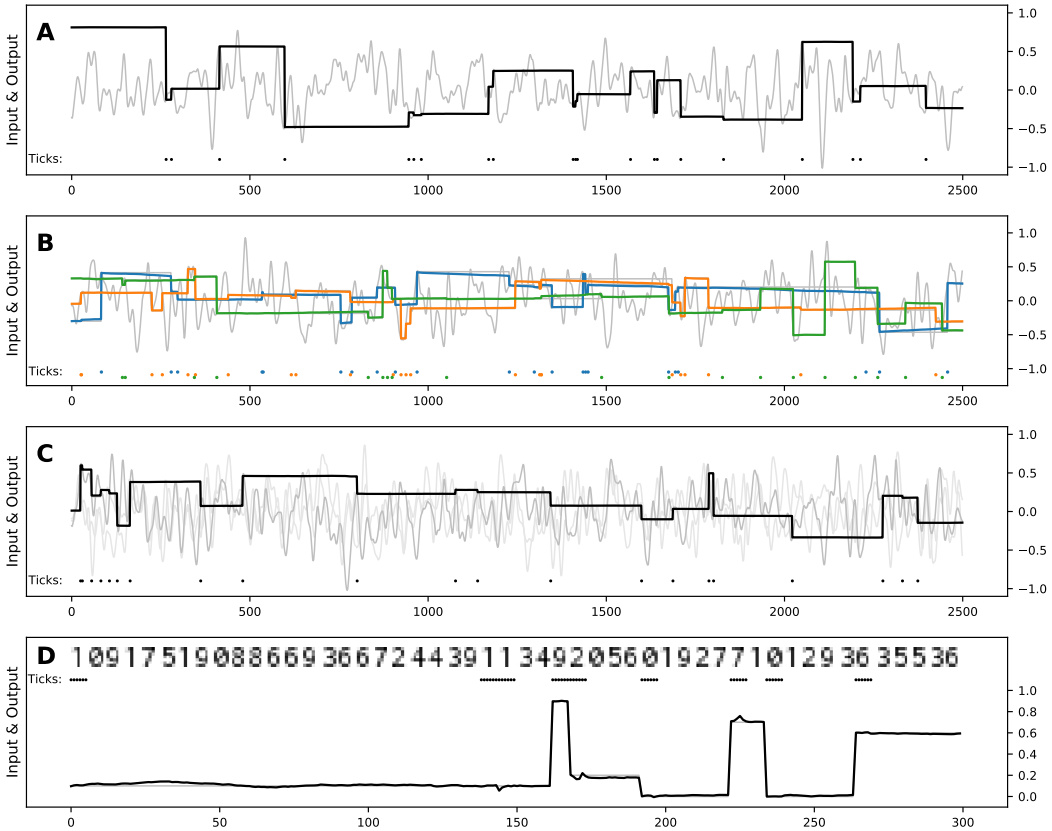


FIGURE 5.6: **Performance of the reservoir model on working memory tasks.** The light gray line is the input signal and the thick black (or colored) one is the output of the model. Dots at the bottom represents the trigger signal (when to memorize a new value). For the digit task, the input containing the value to maintain is shown as a picture instead. **A** *1-value 1-gate scalar task* **B** *1-value 3-gate scalar task* **C** *3-value 1-gate scalar task* **D** *1-value 1-gate digit task*

1-value 3-gate scalar task

We trained the model on the *1-value 3-gate task* using the same protocol as for the 1-value 1-gate task, using a single value input, three input triggers and three corresponding outputs. Since there are now three feedbacks, we divided respective feedback scaling by 3. Figure 5.6B shows that maintaining information simultaneously impacts the performance of the model (illustrative test run). There is no catastrophic effect but performances are clearly degraded when compared to the *1-value 1-gate task*. The RMSE on this test run increased by one order of magnitude and is about $2 * 10^{-2}$. Nevertheless, in the majority of the cases we tested, the error does stay below 10^{-2}. However in a few cases, one memory (and not necessary all) degrades faster.

3-value 1-gate scalar task

We used the same protocol as for the *1-value 1-gate scalar task* but there are now two additional inputs not related to the task and that can be considered as noise. Adding such irrelevant inputs had no effect on the solving of the task as illustrated in Figure 5.6C that shows an illustrative test run of the model. The error in the output is also always kept below 10^{-2} and the RMSE is about the same ($3 * 10^{-3}$). This is an interesting result, because it means the network is not only able to deal with "background noise" at 10^{-4}, but it is also able to deal with noise that has the same amplitude as the input. This is an important property to be considered for the modelling of the prefrontal cortex: being an integrative area, the PFC is typically dealing with multimodal information, many of which being not relevant for the working memory task at hand (Mante et al., 2013).

1-value 1-gate digit task

The model has been trained using 25,000 random integer values between 0 and 9 sampled from an uniform distribution. Each of these values is drawn one after the other onto an image, each character covering 6 columns of the image. The input V consists then of the row of this image. The T signal is sampled similarly as in the *1-value 1-gate task* and then expanded 6 times to fit the transformation of the value to the picture of the values. Which means that trigger lasts 6 time steps. Interestingly, as we show on Figure 5.6D, even if the value to maintain is not explicit anymore, it can still be extracted and maintained. On the test run we show the RMSE is about $4 * 10^{-2}$. It is to be noted that the recognition of a digit is not straightforward and may require a few timesteps before the digit is actually identified. However when the good value is maintained it seems to last, the absolute error stays below 0.05, which is the threshold from which we can distinguish between two values. The reservoir parameters that we found are robust enough to enable not only a pure memory task (i.e. gating), but also a discrimination task (i.e. digit recognition).

Dambre et al. (2012b) demonstrated the existence of a universal trade-off between the non-linearity of the computation and the short-term memory in the information processing capacity of any dynamical systems, including echo state networks. In other words, the hyperparameters used to generate an optimal reservoir for solving a given memory task would not be optimal for a non-linear computation task. Here we see that even if the reservoir is made to memorised a particular value, it is still able to do a non-linear task such as discriminating stream of digits. Pascanu and Jaeger (2011b) initiated the concept of such working memory (WM) units for reservoirs. They processed streams of characters using six binary WM-units to record the deepness of curly brackets that appeared. We made such reservoir-WM-units coupling

more general: from binary to continuous values. Instead of relying on a collection of N binary WM-units to encode N values, or on the maximum encoding of 2^N values, we have shown that a reservoir can use only one WM-unit to encode a continuous value with a good precision.

5.3.3 Robustness

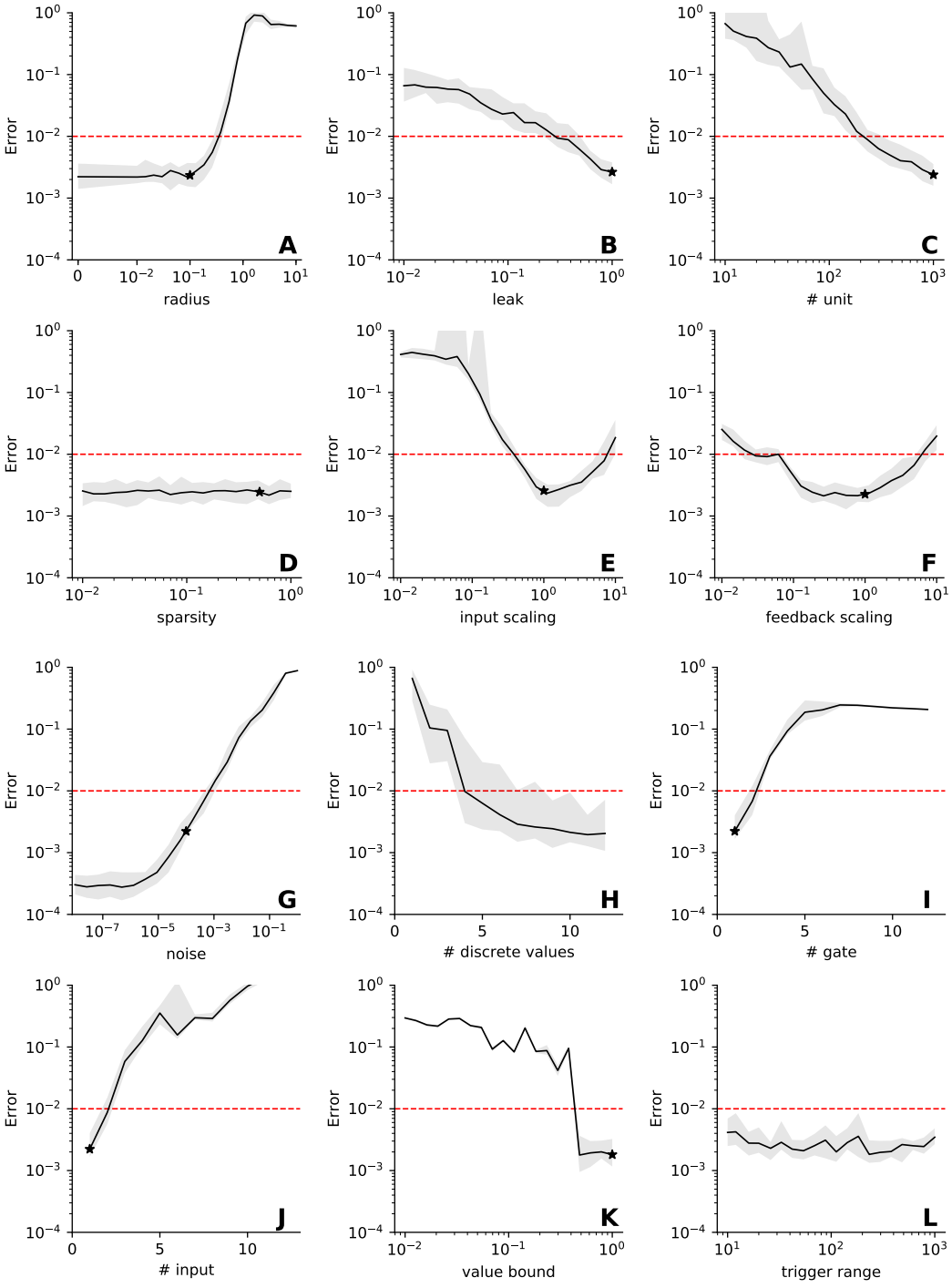


FIGURE 5.7: **Robustness of the model to hyper-parameters** (caption on next page)

FIGURE 5.7: The performance of the model has been measured when a single hyper-parameter varies while all others are kept constant and equal to the value given in table 5.1. For each plot, we ran 20 iterations of the model using different random seeds for initialization. Most of the time (**A-GK-L**), the varying parameter has been split into 20 log-uniformly spread values between the minimum and maximum values. **A-F.** Model hyper-parameters. **G-L.** Task hyper-parameters. **A.** Spectral radius ranging from 0.01 to 10. with an additional 0. **B.** Leak term ranging from 0.01 to 1 (no leak). **C.** Number of units in the reservoir (from 1 to 1000). **D.** Sparsity level ranging from 0.01 to 1.0 (fully connected). **E.** Input scaling ranging from 0.01 to 10. **F.** Feedback scaling ranging from 0.01 to 10. **G.** Noise level inside the reservoir, ranging from 10^{-8} to 1. **H.** Number of discrete values used to train the model, ranging from 1 to 12. **I.** Number of input gates and output channels, ranging from 1 to 12. (1-value n-gate scalar task, RMSE has been divided by the square root of the number of gates) **J.** Number of input values, ranging from 1 to 12. (n-value 1-gate scalar task) **K.** Bound for the input value during training, ranging from 0.01 to 1. **L.** Maximal interval (number of steps) between consecutive ticks during training, ranging from 10 to 1000. $\star$**.** Performance for default parameters (Table 5.1).

We analyzed the robustness of the model first by measuring its sensitivity for each of the hyper-parameters (Figure 5.7A-F), namely: input scaling, feedback scaling, spectral radius, sparsity of the reservoir weight matrix, leak term (α) in equation (5.11) and number of units in the reservoir. We also we measured its sensitivity to task hyper-parameters (Figure 5.7G-L): the noise level (ζ), number of discrete values used during training (when there is a trigger, V is sampled uniformly in uniformly sampled between -1 and 1 discrete values), the temporal delay between successive gating signals in training (T is built sampling its interval between triggers uniformly between 0 and a bound), the bound used to sample the input value in training (V is uniformly sampled between -b and b instead, where b is the bound), the total number of input gates (with a corresponding number of outputs), the number of input values. For most hyper-parameter, we set a minimum and maximum value and pick 20 values logarithmically spread inside this range. For each task and model hyper-parameter we ran 20 simulation instances for 25,000 timesteps and record the mean performance using 2,500 values. Results are shown in Figure 5.7.

First, we can see a non-sensitivity to the sparsity (i.e. minor differences in performances when these parameters vary). Similarly we can see a non-sensitivity to the leak term, input and feedback scaling, as long as they are not too small. It is to be noted that input and feedback scaling should also not be too big. As expected, the performance increases with the number of neurons. Surprisingly, we can note that the performance decreases with the spectral radius. In fact, in supplementary Figure 5.17 we analysed the behavior of the reservoir model with various spectral radii. We show that even with a bigger spectral radius, the reservoir keeps maintaining something relevant but less precise (the segment attractor is slowly degenerating). Globally, the reservoir model is very robust against model hyper-parameters changes as long as it is trained in this condition.

Concerning the task hyper-parameters, one can see in Figure 5.7 that only the trigger range has no impact. This means that, whatever the time elapsed between two triggers, it does not affect the performance. Performance naturally decreases with the increase of the noise level (Figure 5.7**G**), the number of input gates (**I**) or the number of input values (**J**). We can note that the number of discrete values used during training impacts the performance in a very specific way (**H**). Using between 4 to 7 training values, the performance is already good and does not improve further with supplementary training values. This means that even if the reservoir model has been only trained to build few stable points, it is able to interpolate and maintain the other points on the segment attractor. Interestingly, in Figure 5.7**K**, we can see a similar case of interpolation relative to the input value bound x (i.e. the interval $[-x, x]$ on which the output is trained). The performance reached a plateau when the bound values reached 0.5 while the interval used for testing performance is always $[-1, 1]$.

5.3.4 Dynamics

A segment attractor Figure 5.8 shows how the model evolves after having been trained for the *1-value 1-gate task* using different starting positions and receiving no input. This results in the formation of a segment attractor even tough the model was only trained to gate and memorize continuous values.

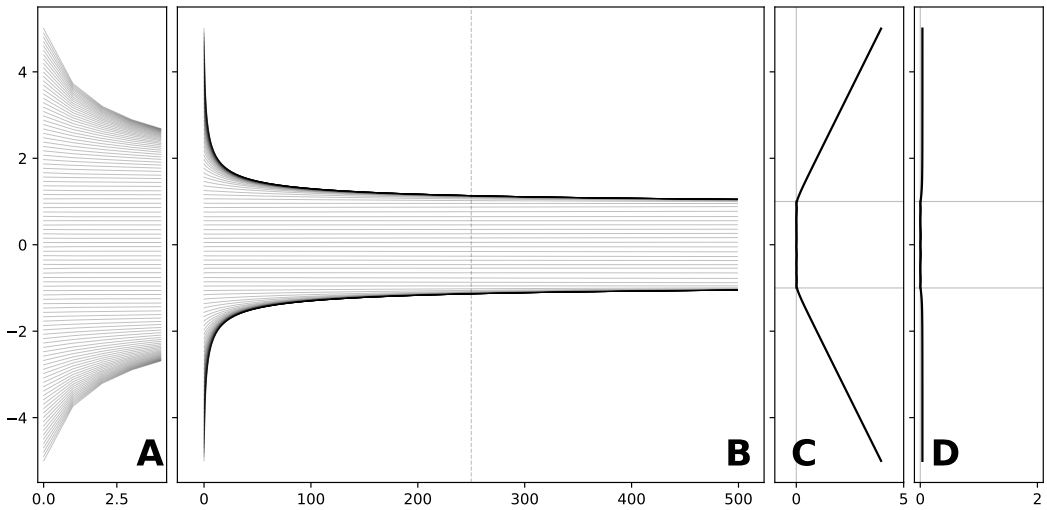


FIGURE 5.8: **An approximate segment attractor** The same model trained on the 1-value 1-gate was tested for 500 iterations without inputs, starting with an initial trigger along with values linearly distributed between -5 and +5. **A** and **B** Output trajectories. Each line corresponds to one trajectory (output value) of the model. **A** is a zoom of **B** on the first few time steps. **C** Measure of how well are maintained the initial triggered value: the absolute difference in the output between initial time and final time. **D** Measure of how stable are the states reached by the reservoir: The maximal absolute difference between states at intermediate time (dashed line) and final time.

If we compute the principal component analysis (PCA) on the reservoir states and look at the principal components (PCs) in the absence of inputs, we can observe (supplementary Figure 5.16) that all the reservoir states are organized along a straight line on the first component (the one which explains most of the variance) and each point of this line can be associated with a corresponding memory value. Interestingly enough, there are points on this line that correspond to values outside the $[-1;1]$ range, i.e. values for which the model has not been trained for. However, these points are not stable and an any dynamics starting from these points converge towards the points associated to the values 1 or -1 (see Figure 5.8).

***V*-like and *M*-like neurons** Similarly to the minimal model, in the absence of input, the inner dynamic of the reservoir model is a combination of both sustained and highly variable activities. More precisely, in the *1-value 1-gate task* we notice two types of neurons that are similar to neurons $X1$ and $X3$ in the reduced model: (1) neurons which solely follows the input V (i.e. V-like

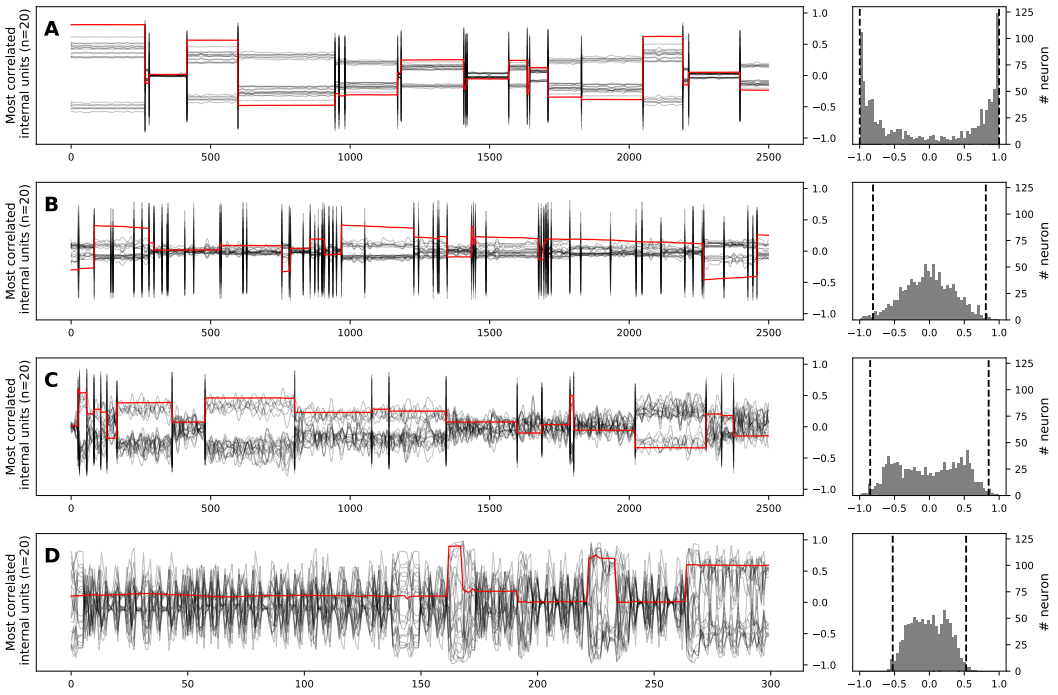


FIGURE 5.9: **Most correlated reservoir units displaying various degrees of maintained activity.** Left: in black the activities of the 20 neurons the most correlated with the output, in red the output of the model. Right: histogram of the correlation between the neurons and the output; y-axis represents the number of neurons correlated with the output produced by the model within a correlation bin. Dashed lines indicate the 20th most correlated and most anti-correlatted neurons with the output. **A** *1-value 1-gate scalar task* **B** *1-value 3-gate scalar task* **C** *3-value 1-gate scalar task* **D** *1-value 1-gate digit task*. Most correlated reservoir units do not necessarily show clear sustained activity: we can see a degradation of sustained activities from **A** to **D** according to the different tasks. Thus, sustained activity is not mandatory to perform working memory tasks.

neurons), and (2) neurons that mostly follow the output M (i.e. M-like neurons), respectively. We also notice that $M(|w| \leq 0.1)$-like neurons average activity is linearly linked with the M value and fluctuate around this mean activity according to the input V. In Figure 5.9 we show M-like neurons for the different tasks. These neurons were found by taking the most correlated neurons with the output M[3]. From Figure 5.9**A** to **D** we see that the M-like neurons link with M output goes weaker and weaker: the average sustained activity goes from nearly flat to highly perturbed.

This "degradation" of sustained activity is explained by the change in the distribution of correlations of the whole reservoir population with M output: in Figure 5.9 **(right)** we see that the correlation with M output is quickly shrinking from **A** to **D**. For the *1-value 1-gate task* (5.9A) mostly all neurons stay at the same value while maintaining the memory. However, when more values have to be maintained (Figure 5.9B) or when more inputs are received (Figure 5.9C), most of the activities do not stay at the same value anymore while maintaining the memory. In fact, in the *1-value 1-gate digit task*, neurons do not display a sustained activity at all (Figure 5.9D). Interestingly, similar behavior (no sustained activity) can be obtained by lowering the feedback scaling (supplementary Figure 5.18) or by enforcing the input weights to be big enough (supplementary Figure 5.20). More formally, when there is no trigger, the activity of the neurons can be rewritten as $\tanh(aX + bM)$. The two proposed modifications make the ratio $\frac{a}{b}$ bigger and eventually when $a \gg b$, $\tanh(aX + bM) \approx \tanh(aX)$. Consequently, $\tanh(aX)$ is highly correlated with X as aX stays bounded between -1 and 1, and does not depend of M. Similarly, when $a \ll b$, $\tanh(aX + bM) \approx \tanh(bM)$ which is in turn highly correlated with M for the same reasons.

Linear decoder To go further in the understanding of the role played by sustained activities, we wanted to know how much of these sustained activities were necessary to decode the output memory M. For the *1-value 1-gate task*, we trained a separate decoder based on a subsample of the reservoir population. We increasingly subsampled neurons based on three conditions: either by choosing the most correlated one first, the least correlated one first or just randomly selecting them. In Figure 5.10 we can see two interesting facts. First, there is no need of the whole reservoir population to decode well enough the memory M: taking a hundred neurons among one thousand is sufficient. Second, if we take enough neurons, there is no advantage in taking the most correlated one first, random ones are enough. Surprisingly, it seems better to rely on randomly selected units than most correlated ones. This suggests that randomly distributed activities contained more information than just most correlated unit activities and offers a complementary perspective when comparing it to Rigotti et al. (2013b) decoding of neural population recorded in monkeys: when task-related neurons are not kept for decoding, information (but less accurate) can still be decoded.

[3]Because a trigger input has a substantial influence on the reservoir states, we made the categorization by ignoring the time steps when there is a trigger.

5.3.5 Equivalence between the minimal and the reservoir model

In order to understand the equivalence between the minimal and the reservoir model, it is important to note that there are actually two different regimes as shown in Figure 5.11. One regime corresponds to the absence of a trigger (T=0) and the other regime corresponds to the presence of a trigger (T=1). When there is no trigger (T=0), the activities of X_1 and X_2 compensate each other because they are in quasi-linear regime (b being very small) and their summed contribution to the output is nil.

In the reservoir model, we can actually identify an equivalent population by discriminating neurons inside the reservoir based on the strength of their input weight relatively to V. More formally, we define R_{12} as *the group of neurons whose input weight from V (absolute value) is greater than* 0.1. Figure 5.12 (panel H) shows that the summed output of these neurons is quasi nil while the complementary population R_3 is fully correlated with the output and is thus equivalent to the X_3 unit in the minimal model.

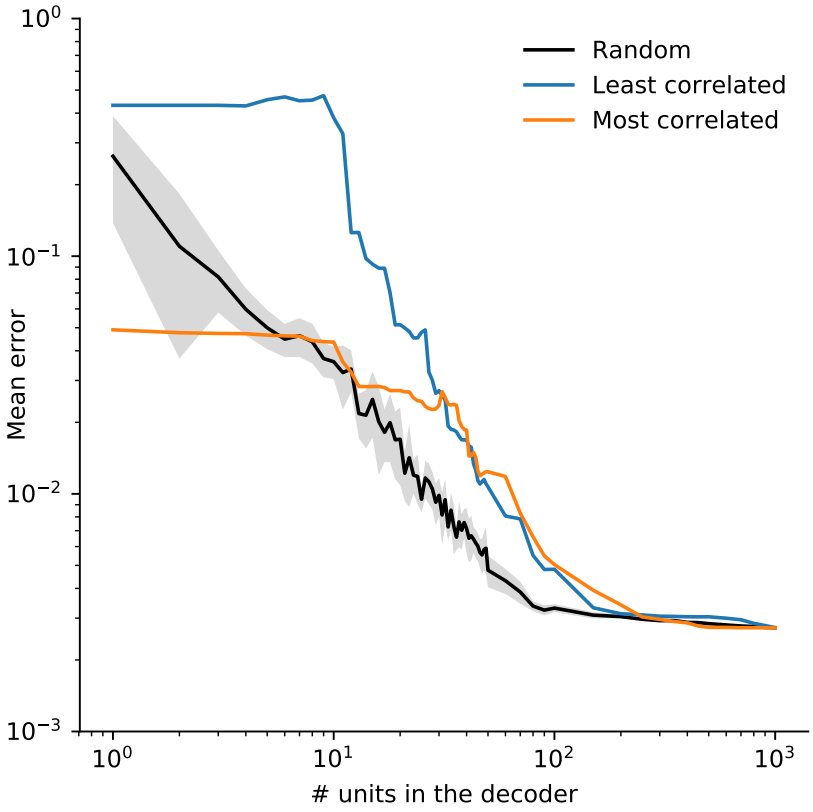


FIGURE 5.10: **Performance of variable size linear decoders.** After training, we ran the model on the training set and recorded all the internal states. We selected a subset (from 1 to 1000 units) of these internal states and find the associated best linear decoder on the training data set. Finally, we measured the performance of these decoders on the training sets. Units composing a subset have been sampled either using the least or the most correlated units (with the output) or simply randomly. Performance for subsets of size 1000 is equal to the performance of the original model.

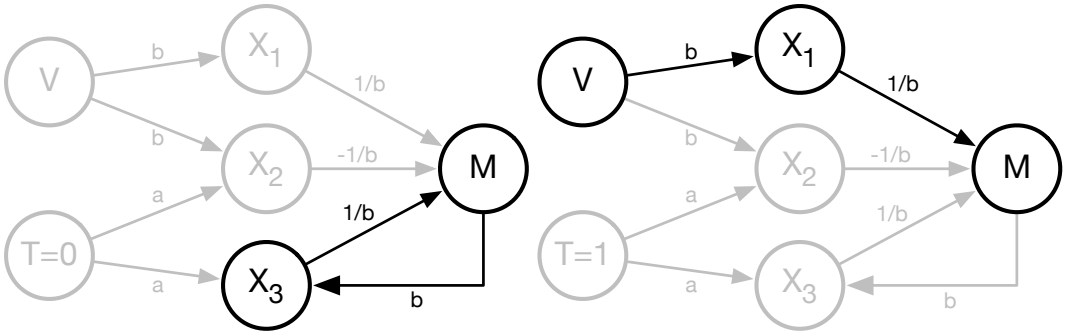


FIGURE 5.11: The two regimes of the minimal model depending on the absence (left) or the presence (right) of a trigger T (0 or 1).

Symmetrically, in the presence of a trigger (T=1), the activities of X_2 and X_3 compensate each other because they are in saturated regime (a being very large) and their summed contribution to the output is nil. We can again identify an equivalent population in the reservoir model by discriminating neurons inside the reservoir based on the strength of their input weights relatively to both T and V. More formally, we define R_{23} as *the group of neurons whose input weight from T (absolute value) is greater than* 0.05 *and whose input weight from V (absolute value) is smaller than* 0.1. Figure 5.12 (panel J) shows that the summed output of these neurons is quasi nil while the complementary population R_1 is fully correlated with the input V and is thus equivalent to the X_1 unit in the minimal model.

We can identify R_2 by taking the intersection of R_{12} and R_{23} whose activity is similar to X_2 (see panel L in Figure 5.12). Consequently, we have identified in the reservoir disjoint sub-populations that are respectively and collectively equivalent to activity of X_1, X_2 and X_3 in the minimal model. Table 5.2 quantifies this equivalence using simple correlations. To explore further this equivalence, we also conducted comparative lesion studies between the two models (see Supplementary Materials). However, with the original set of parameters for the reduced model ($a = 1000, b = 0.001$), lesioning X_2 or X_3 makes the reduced model to behave in a degraded and *extreme* mode: outputs range from 0 to ± 1000 whenever there is a trigger and it makes the comparison with the reservoir difficult (see Supplementary Figure 5.21). By choosing an alternative set of parameters ($a = 1000, b = 1$ that incidently makes the reduced model unable to sustain memory), we can show a strong correlation with the reservoir when X_1/R_1 (resp. X_2/R_2, X_3/R_3) are silenced (see Supplementary Figure 5.22) and this further tighten the relationship between the two models.

Populations	**Output correlation**
X1 / R1 (T=1)	0.9996
X2 / R2 (T=0)	0.9997
X3 / R3 (T=0)	0.9997

TABLE 5.2: Correlation between respective subpopulations in the reduced and reservoir models. Output is restricted to the contribution of the considered subpopulation.Correlations are computed only at time relevant for the subpopulation ($T = 0$ or $T = 1$).

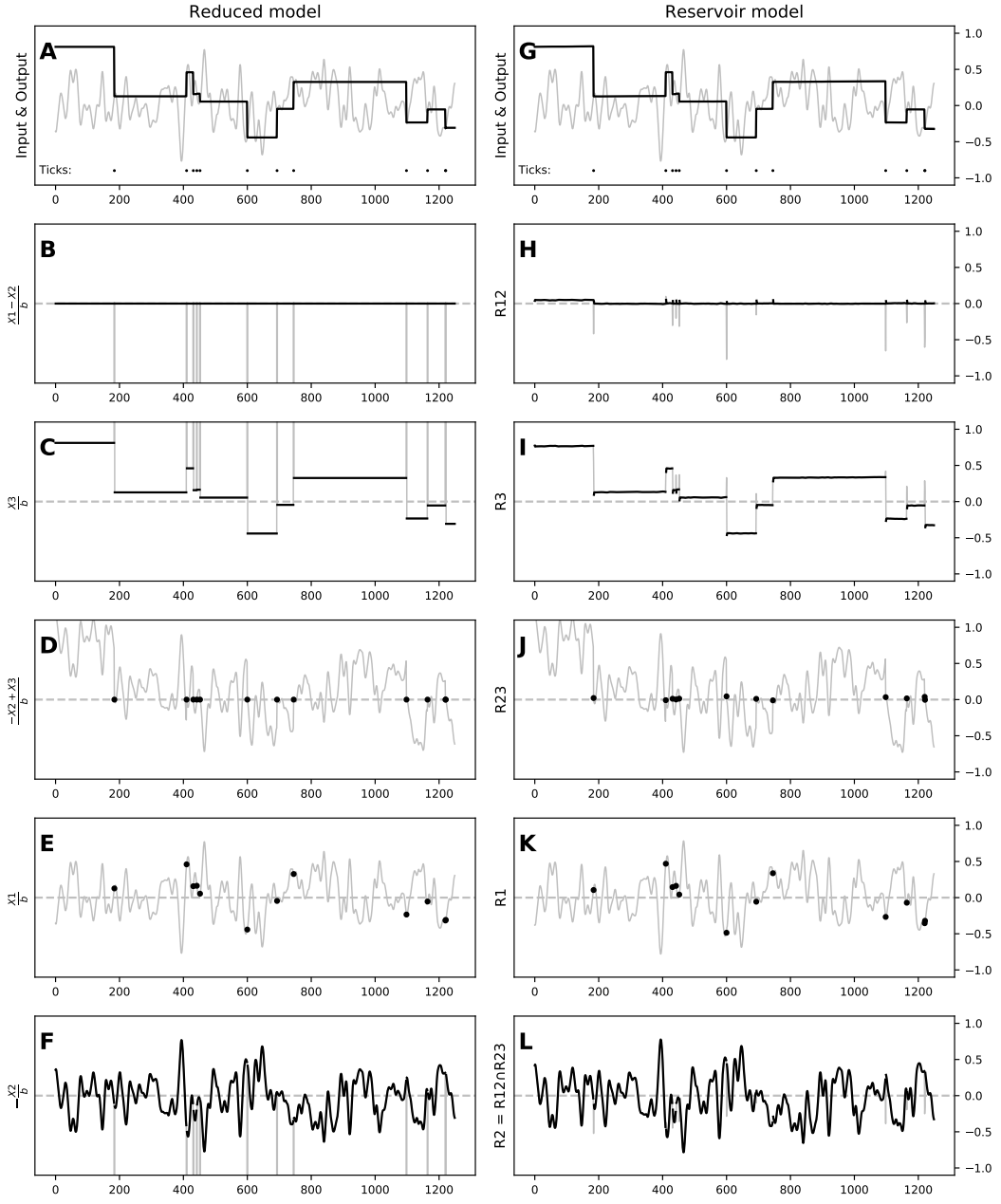


FIGURE 5.12: **Side by side comparison of the minimal (left) and full (right) models. A & G** Output of the two models respectively. **B & H** Group of neurons whose sum of activity is nil between triggers. **C & I** Group of neurons whose sum of activity is equal to the output between triggers. **D & J** Group of neurons whose sum of activity is nil during triggers. **E & K** Group of neurons whose sum of activity is equal to the input during triggers. **F & L** Group of neurons whose sum of activity is equal to the input between triggers.

5.4 Discussion

In computational neuroscience, the reservoir computing paradigm (RC) (Jaeger, 2001a; Maass, Natschläger, and Markram, 2002; Verstraeten et al., 2007), originally proposed independently by Dominey (1995), and Buonomano and

Merzenich (1995)[4], is often used as a model of canonical microcircuits (Maass, Natschläger, and Markram, 2002; Hoerzer, Legenstein, and Maass, 2012; Sussillo, 2014). It is composed of a random recurrent neural network (i.e. a reservoir) from which readout units (i.e. outputs) are trained to linearly extract information from the high-dimensional non-linear dynamics of the reservoir. Several authors have taken advantage of this paradigm to model cortical areas such as PFC (Hinaut and Dominey, 2013; Mannella and Baldassarre, 2015; Hinaut et al., 2015; Enel et al., 2016b) because most of the connections are not trained, especially the recurrrent ones. Another reason to use the reservoir computing paradigm for PFC modelling is because PFC also hosts high-dimensional non-linear dynamics (Rigotti et al., 2013b). RC offers a neuroanatomically plausible view of how cortex-to-basal-ganglia (i.e. cortico-basal) connections could be trained with dopamine: the reservoir plays the role of a cortical area (e.g. trained with unsupervised learning), and the read-out units play the role of basal ganglia input (i.e. striatum).

However, in many dynamical systems, reservoirs included, there exists a trade-off between memory capacity and non-linear computation (Dambre et al., 2012b)[5]. This is why some studies have focused on reservoirs with dedicated readout units acting as working memory (WM) units (Hoerzer, Legenstein, and Maass, 2012; Pascanu and Jaeger, 2011b; Nachstedt and Tetzlaff, 2017). These WM units have feedback connections projecting to the reservoir and are trained to store binary values that are input-dependent. This somehow simplifies the task and enables the reservoir to access and use such long-time dependency information to perform a more complex task, freeing the system from constraining reservoir short-term dynamics. Such ideas had already some theoretical support, for instance Maass, Joshi, and Sontag (2007) showed that with an appropriate readout and feedback fonctions, readout units could be used to approximate any k-order differential equation. Pascanu and Jaeger (2011b) used up to six binary WM units to store information in order to solve a nested bracketing levels task. Using a principal component analysis, they showed that these binary WM units constrain the reservoir in lower dimensional "attractors". Additionally, Hoerzer, Legenstein, and Maass (2012) showed that analog WM units (encoding binary information) also drive the reservoir into a lower dimensional space (i.e. 99% of the variability of the reservoir activities are explained by fewer principal components). More recently, Strock, Rougier, and Hinaut (2018), Beer and Barak (2019) used such WM units in order to store analog values (as opposed to binary ones) in order to build a *line attractor* (Seung, 1996; Sussillo and Barak, 2013b). In particular, Beer and Barak (2019) explored how a line attractor can be built online, by comparing FORCE (Sussillo and Abbott, 2009) and LMS algorithms, using a WM unit to maintain a continuous value in *the absence* of input perturbations.

In that context, the minimal model of three neurons we have proposed helps

[4]Earlier formulations of very similar concepts can be found in (Jaeger, 2007).

[5]For reservoirs this trade-off depends on the hyper-parameters (HP) chosen: some HP sets will give more memory, others more computational capacity (Legenstein and Maass, 2007).

to understand the mechanisms that allows a reservoir model to gate and maintain scalar values, in *the presence* of input perturbations, instead of just binary values. As explained previously, this minimal model exploits the non-linearity and the asymptotic behavior of the three tanh units and mimics a select operator between the input signal and the output. In the case of the reservoir model, there is no precise architecture or crafted weights, but this is compensated by the size of the population inside the reservoir along with the training of the output weights. More precisely, we have shown that virtually any population of randomly connected units is able to maintain an analog value at any time and for an arbitrary delay. Taking advantage of the non-linearity of the neuron transfer function, we have shown how such population can learn a set of weights during the training phase, using only a few representative values. Given the random nature of the architecture and the large set of hyper-parameters, for which the precision of the output remains acceptable, this suggests that this property could be a structural property of any population that could be acquired through learning. To achieve such property we mainly used offline training in our analyses (for efficiency reasons), but we have shown that it also works with online FORCE learning (see supplementary materials and Figure 5.14 and 5.15).

We have shown that the reservoir model behavior is similar to the minimal model with the presence of two "macro states" that are implemented by compensatory clusters. In a nusthell, this working memory is using two distinct mechanisms: a selection mechanism (i.e. a switch), and a line attractor. Such mechanims have been also reported in a fully trained recurrent neural network with back-propagation (Mante et al., 2013). The authors proposed a context-dependent selective integration task and showed that "the rich dynamics of PFC responses during selection and integration of inputs can be characterized and understood with just two features of a dynamical system the line attractor and the selection vector, which are defined only at the level of the neural population". However in our case, not only did we rely on the analysis of the dynamical system to understand the behavior of the system, but we were able to design a minimal model implementing these mechanisms and show these same mechanisms are also present in the reservoir model but in a distributed way.

Finally, one important feature of the model is that it is actually an open system and as such, it is continuously under the direct influence of external activities. More precisely, the model is able to retain an information when the gate is closed, but this *closed gate* corresponds to a functionnal state rather than a physical state where input signals would have been blocked and information continues to enter the system. This is illustrated quite clearly when one looks at the internal activities inside the reservoir: a large number of neurons are directly (and only) correlated with the input signals. This has consequences for the analysis of the dynamics of the population: this population is partly driven by the (working memory) task and partly driven by uncorrelated external activities. If we go back to biology, this makes perfect sense: a population of neurons is never isolated from the rest of the brain.

When studying population electrophysiological recordings, we have to keep in mind that such activities can be fully uncorrelated with the task we are observing. This might be one of the reasons for the variability of hypotheses about working memory encoding mechanisms.

5.5 Supplementary

5.5.1 Reduced model

The effect of parameters a and b are of distinct nature as illustrated in Figure 5.13. b controls the time the memory takes to fade to zero whereas a controls how the model will mix the previous output and the new desired one. Interestingly, the time the information takes to fade out to zero is not directly linked to a time constant of neurons. Moreover, the mixing factor can actually be computed for b small enough and is equal to $\tanh(a)^2$. Consequently, by feeding T with a (instead of 1) and fixing the input weights coming from T to 1 (instead of a), one can obtain a model which mimics the update mechanism of a Gated Recurrent Unit (GRU) but with regular neurons.

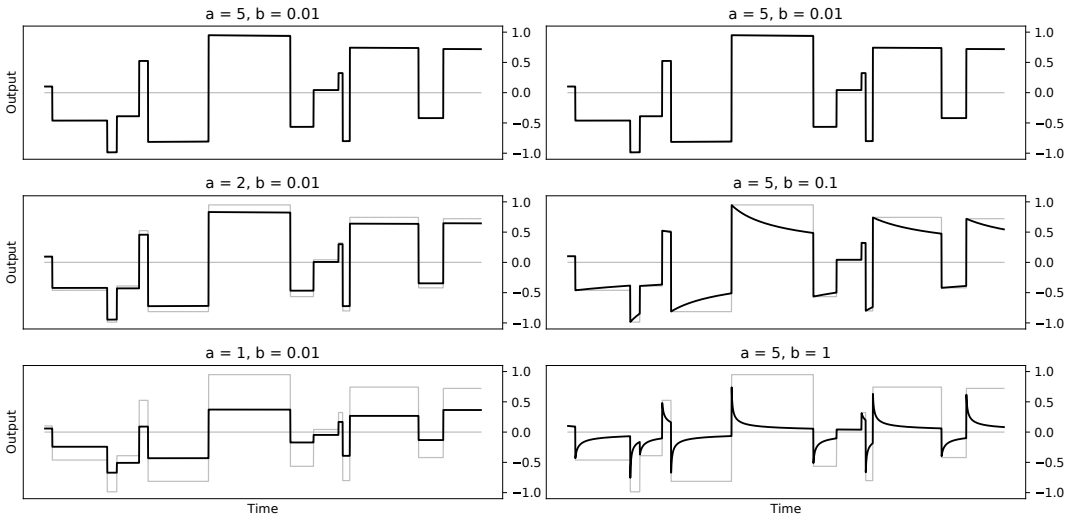


FIGURE 5.13: **Influence of parameters a and b on the reduced model.** The parameter b controls the amount of feedback fed to the model. If b is too high ($b = 0.1$, $b = 1.0$), the memory quickly fade to zero after a few timesteps. The parameter a controls the gate behavior and the ratio of the memorized and the new value that will constitute the new memorized value. If a is too low ($a = 1$, $a = 2$), there is a systematic undershoot.

5.5.2 Online learning

We tested online learning by using FORCE (Sussillo and Abbott, 2009) described by the following update equations:

$$W_{out}[n] = W_{out}[n-1] - e_{-}[n]P[n]x[n] \tag{5.13}$$

$$P[n] = P[n-1] - \frac{P[n-1]x[n]x^T[n]P[n-1]}{1 + x^T[n]P[n]x[n]} \tag{5.14}$$

where $e_{-}[n]$ represents the error made at time n if there were no update of W_{out} and $P[n]$ is a squared matrix computing an online inverse of $XX^T - \alpha I$, α a regularization term and P initialized to $P[0] = \frac{I}{\alpha}$.

In Figure 5.14 and 5.15 , one can see that the behavior obtain with online learning is similar to offline learning.

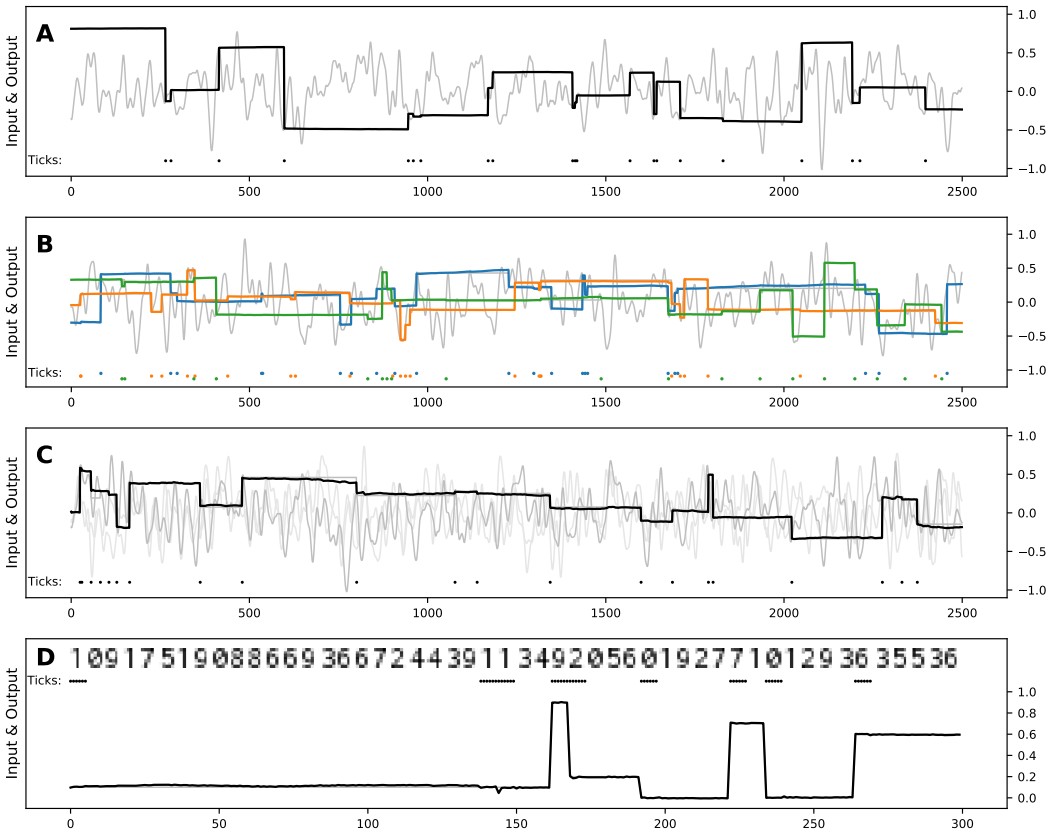


FIGURE 5.14: **Performance of the reservoir model on working memory tasks with online FORCE learning.** Same training/testing protocol than for Figure 5.9. The regularization parameter α has been fixed to 0.0001. The light gray line is the input signal and the thick black (or colored) one is the output of the model. Dots at the bottom represents the trigger signal (when to memorize a new value). For the digit task, the input containing the value to maintain is shown as a picture instead. **A** *1-value 1-gate scalar task* **B** *1-value 3-gate scalar task* **C** *3-value 1-gate scalar task* **D** *1-value 1-gate digit task.*

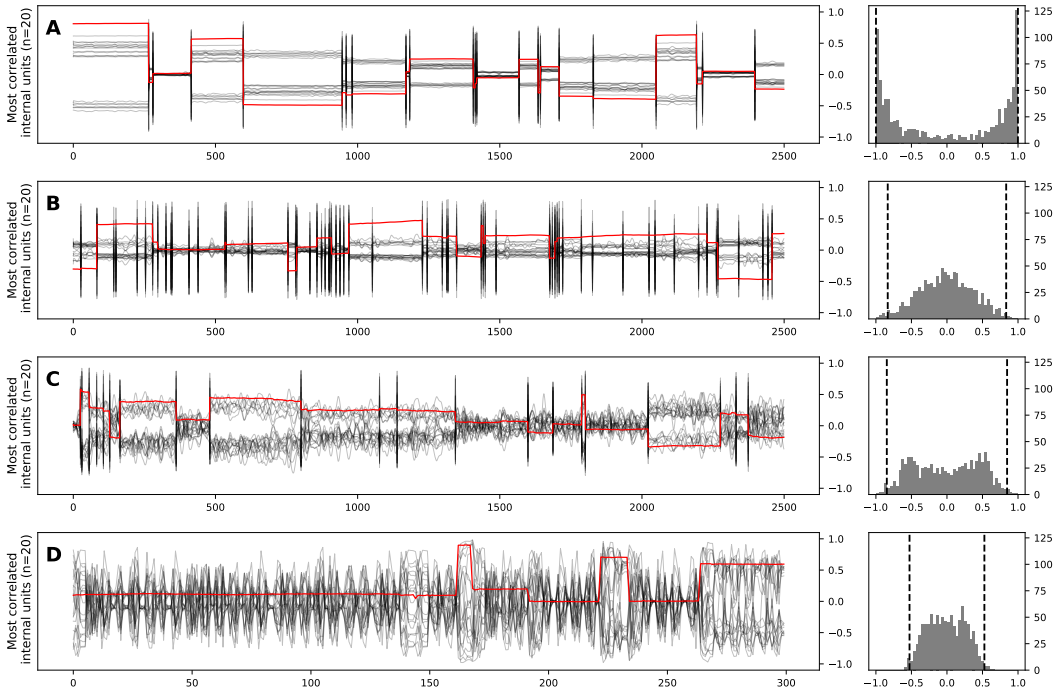


FIGURE 5.15: **Most correlated reservoir units displaying various degrees of maintained activity with online FORCE learning.** Same training/testing protocol than for Figure 5.9. The regularization parameter α has been fixed to 0.0001. (Left) in black the activities of the 20 neurons the most correlated with the output, in red the output of the model. (Right) histogram of the correlation between the neurons and the output; y-axis represents the number of neurons correlated with the output produced by the model within a correlation bin. Dashed lines indicate the 20th most correlated and most anti-correlatted neurons with the output. **A** *1-value 1-gate scalar task* **B** *1-value 3-gate scalar task* **C** *3-value 1-gate scalar task* **D** *1-value 1-gate digit task*. Most correlated reservoir units do not necessarily show clear sustained activity: we can see a degradation of sustained activities from **A** to **D** according to the different tasks. Thus, maintained activity is not compulsory to perform working memory tasks.

5.5.3 More on the segment attractor

In Figure 5.16 we performed a principal component analysis on the reservoir state obtained in Figure 5.8. We can note two interesting facts: (1) The activity evolves in a very low dimensional space, the first component explains more than 99% of the variance by itself, and (2) this component is linked to the memory maintained, linearly between -1 and 1 and non-linearly outside. With both combined together we can say that the segment attractor we built is actually a straight line.

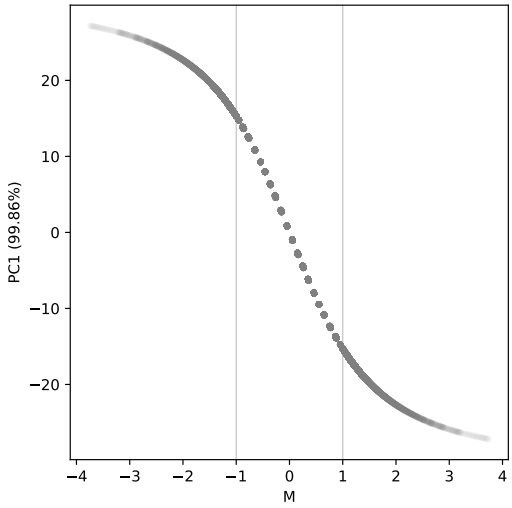


FIGURE 5.16: **The line attractor.** The reservoir state and output analyzed are the one of Figure 5.8. x-axis: output (memory). y-axis: first principal component of the reservoir state.

5.5.4 Influence of the spectral radius on the segment attractor

By watching in Figure 5.17 how the segment attractor is evolving against the spectral radius, we can better explain its influence on the solving of the 1-value 1-gate task. The smaller the spectral radius the more precise is the approximation of the segment attractor. For a 0.1 spectral radius the approximate segment attractor seems continuous. The difference is quasi-null for starting values in [-1,+1] while for values outside this range, the difference corresponds to the difference with the nearest bound of the [-1,+1] segment. For a 0.5 spectral radius we clearly can see that the segment attractor is discretized. In the very first step the outputs concentrates around discretized value between -1 and 1 and is kept constant. In the extreme case for a 1.0 spectral radius it is discretized into two points corresponding to -1 and 1.

5.5.5 Killing neurons having a sustained activity

In Figure 5.18 we show how the neurons are correlated with the output (M), the value (V) and the trigger (T) for different feedback scaling. Because some irregularities come with triggers, we removed the trigger time steps while computing the correlation with the output and the value. We can note that the profile of neurons seems to change according to the feedback scaling. When it is high (1.0), there is mostly neurons following the output, when it is low (0.1) few neurons continue to follow the output while most of them vary according to the value they receive as input. To better identify the neurons which are following the output we looked at where were located the weights of these neurons in Figure 5.19. We can note that the most correlated neurons with the output are mainly the ones with relatively small input weights coming from the value. To go further with this idea we changed the sampling of

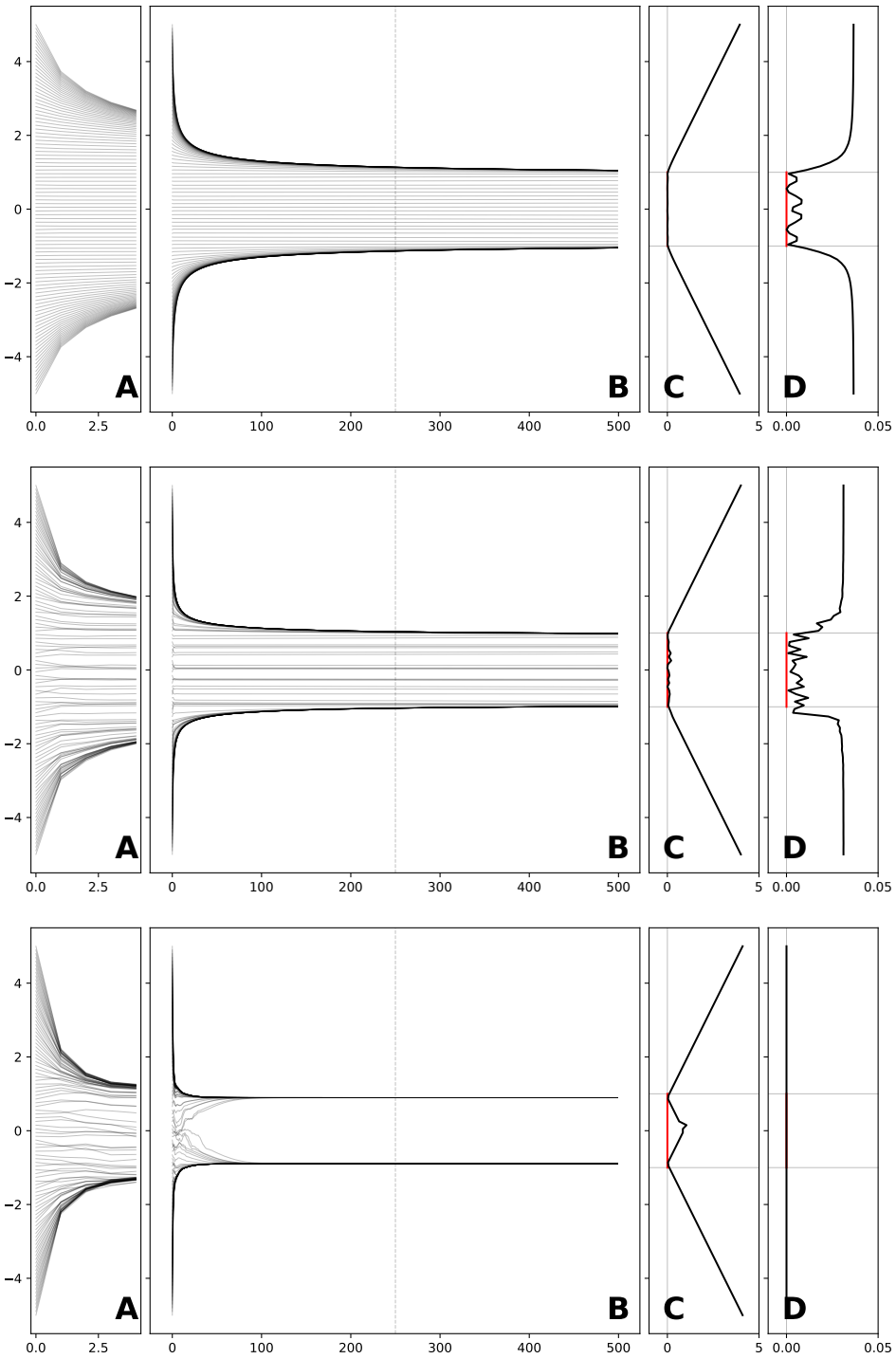


FIGURE 5.17: **Influence of spectral radius on approximate line attractor** The same trained models were tested for 500 iterations without inputs, only starting with an initial trigger along with values linearly distributed between -5 and +5. First line: Spectral radius 0.1. Second line: Spectral radius 0.5. Third line: Spectral radius 1.0. **A** and **B** Output trajectories. Each line corresponds to one trajectory (output value) of the model. **A** is a zoom of **B** on the first few time steps. **C** Asolute difference in the output between initial time and and final time. **D** Maximal absolute difference in the neurons between intermediate (dashed line) and final time.

the input weights to remove small values. In practice the sampling was done uniformly in [-1,-0.5]∪[0.5,1.0] instead. The behavior of this modified model is shown in Figure 5.20. We can note that the model is still able to perform the task relatively well, just a little worse than before, but there is no more neuron displaying a sustained activity. Interestingly, in Figure 5.19, the neurons the most correlated with the trigger displays an intringuing property, their output weights seems to be correlated to their feedback weights.

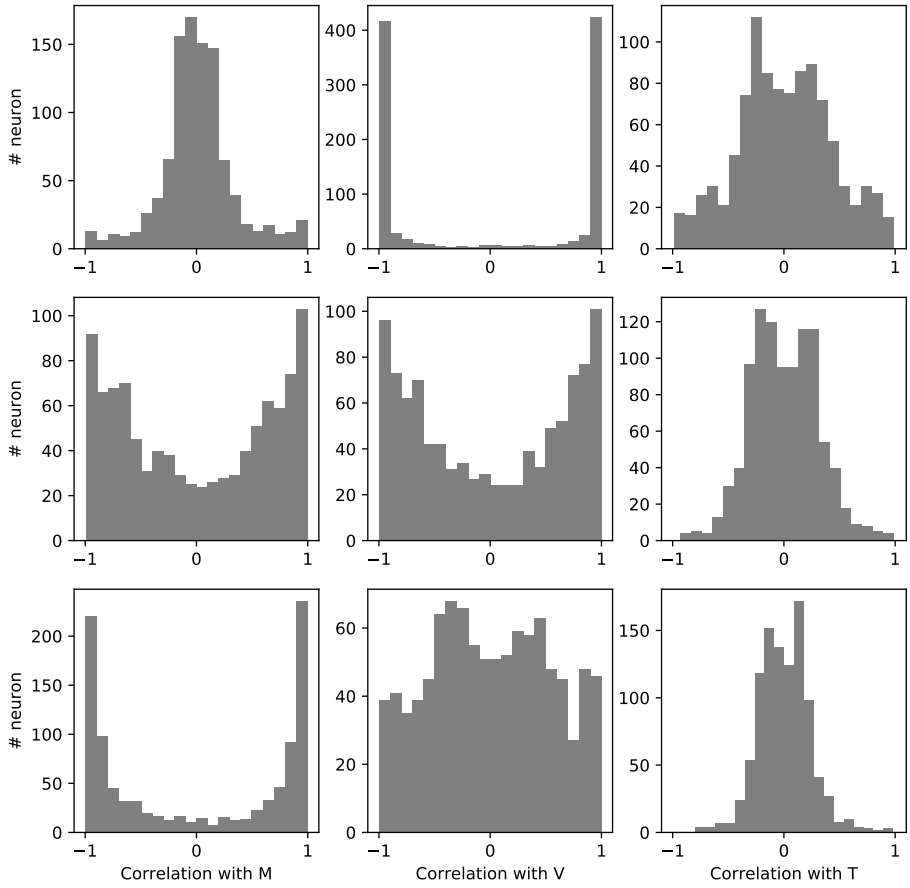


FIGURE 5.18: **Influence of feedback scaling on maintenance of neurons** First line: Feedback scaling 0.1. Second line: Feedback scaling 0.5. Third line: Feedback scaling 1.0. First column: Correlation with M. Second column: Correlation with V. Third column: Correlation with T.

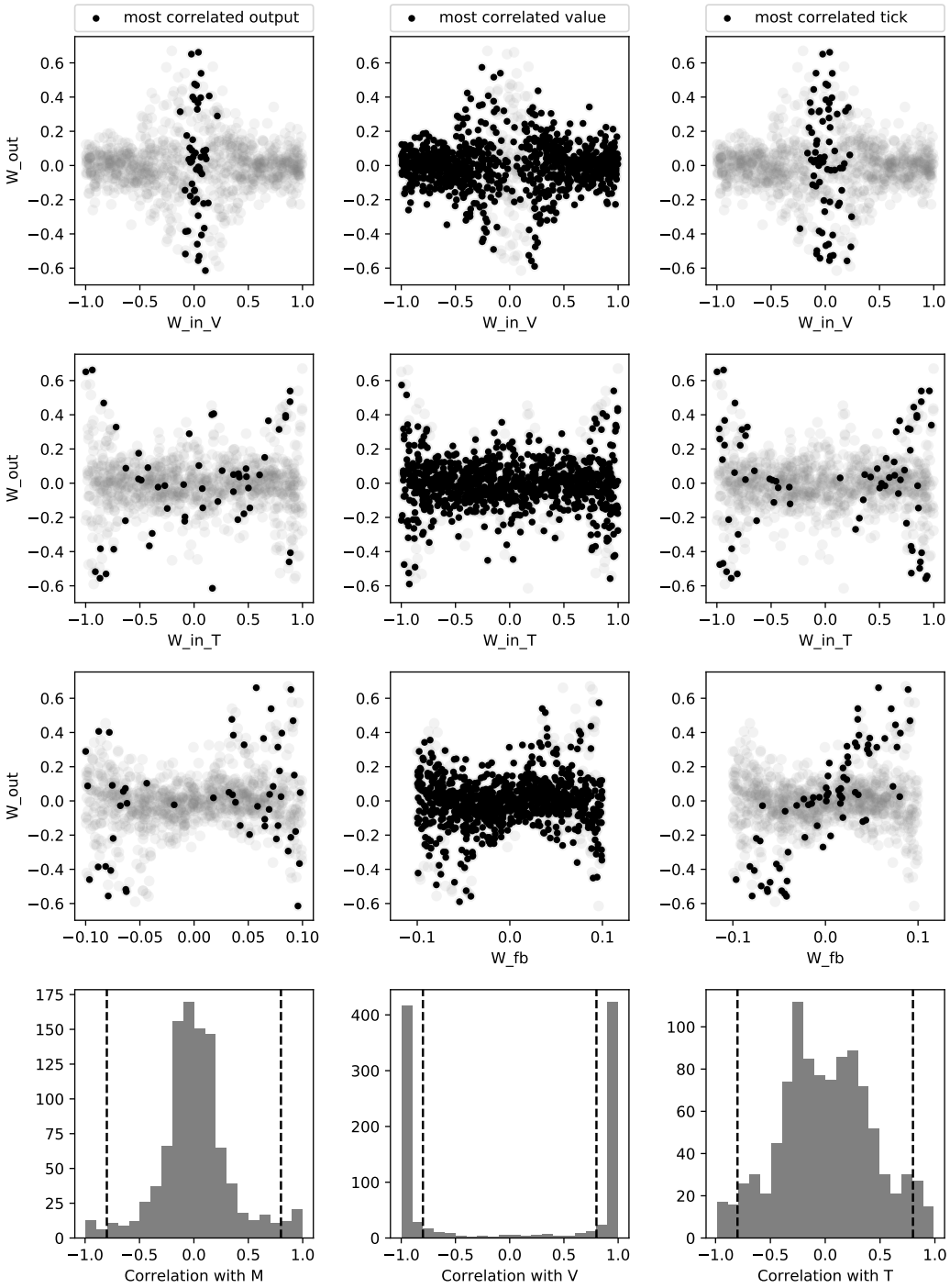


FIGURE 5.19: **Link between weights and behavior of the neurons** The weights of the neurons the most correlated with either M, V or T are highlighted. First column: M. Second column: V. Third column: T. The three first lines show the link between the output weights and some other weights. First line: weights coming from V. Second line: weights coming from T. Third line: weights coming from M. The fourth line shows the correlation with the quantity used to highlight. The threshold used to decide which are the most correlated is shown in dotted black.

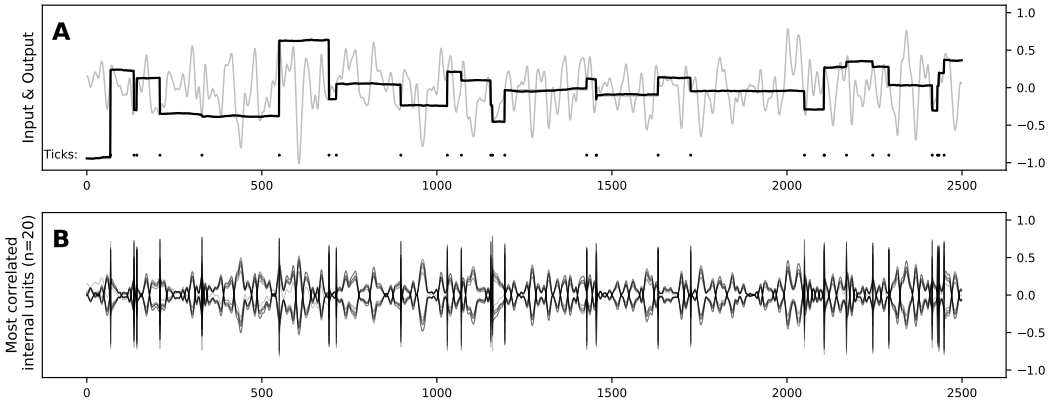
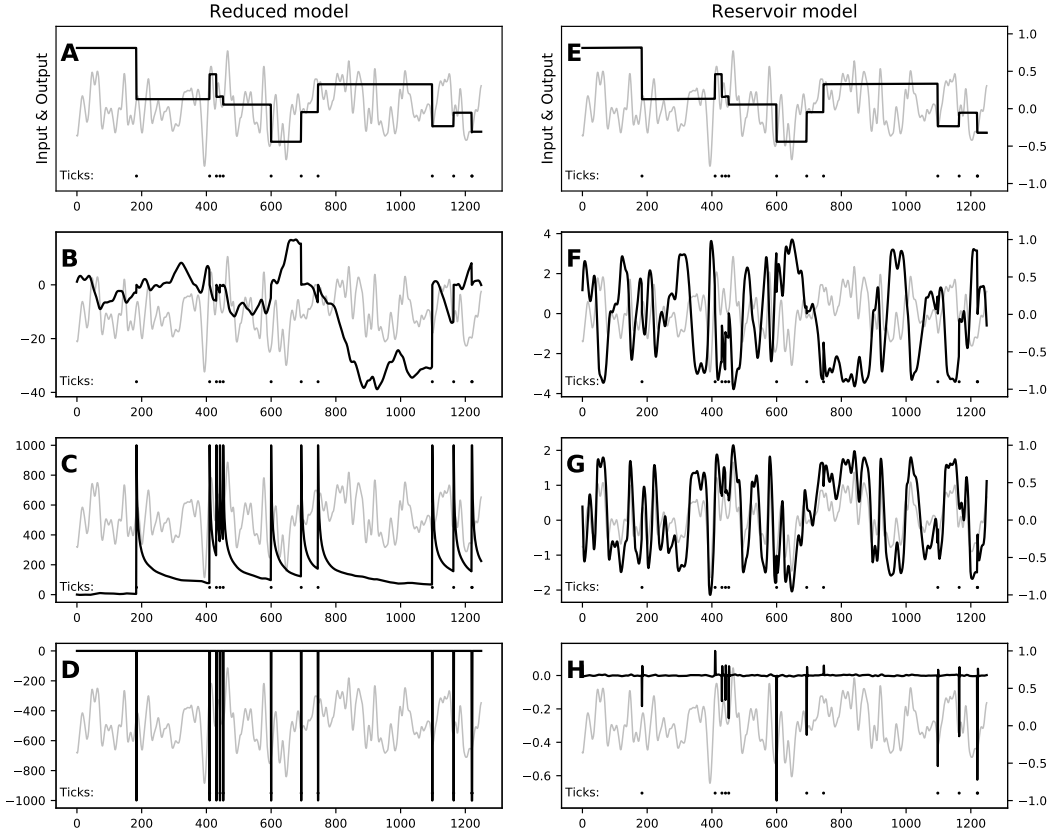


FIGURE 5.20: **Performance of the model with high input value weights on the 1-gate task. A** The light gray line is the input signal and the thick black one is the output of the model. Black dots at the bottom represents the trigger signal (when to memorize a new value). **B** Activity of 20 units that are the most correlated with the output.

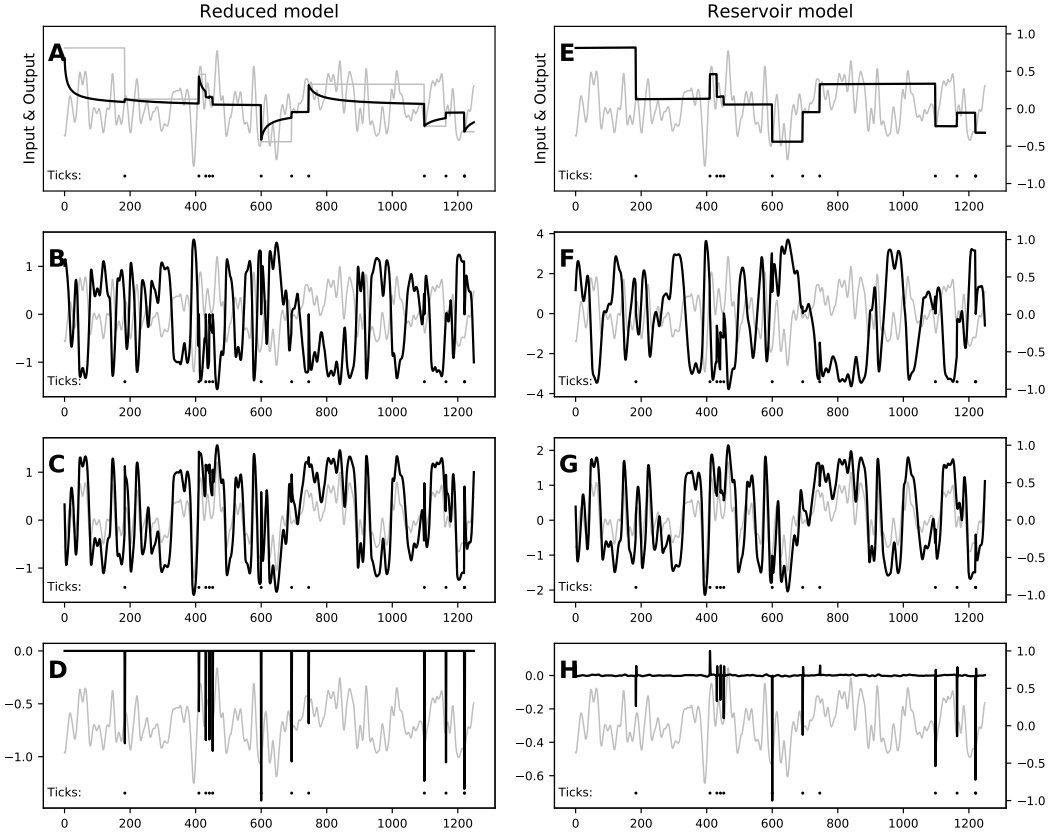
5.5.6 Comparative lesion studies

To explore further the link we established between the reservoir and the reduced model, we conducted comparative lesion studies by forcing a given population output to be 0 (anytime) in both models. We first compared the original reduced model ($a = 1000, b = 0.001$) with the reservoir model. However, since b is very small, and since X_2 and X_3 saturates, their contribution to the output become very large in the presence of a trigger, making the comparison irrelevant (see Figure 5.21). We thus considered an alternative reduced model ($a = 1000$, $b = 1$) to prevent the output to become very large. This alternative intact model is not able to sustain the working memory anymore but the output lesioned model is now strongly correlated with the output of the reservoir model (see Figure 5.22). This further tighten the link between the different subpopulation X_1/R_1, X_2/R_2 and X_3/R_3 even though it is only valid for this specific set of parameters.



Silenced populations	Output correlation
X1 / R1 (T=0)	0.2995
X2 / R2 (T=0/1)	0.1400
X3 / R3 (T=0/1)	0.7549

FIGURE 5.21: **Side by side comparison of lesions in the minimal (left) and full (right) models.** $a = 1000$, $b = 0.001$. **A & E** No lesion. **B & F** X1-R1 lesioned. **C & G** X2-R2 lesioned. **D & H** X3-R3 lesioned. Bottom table: Correlation between output of reduced and reservoir model when one of the component is silenced. Right y-axis: Input (gray) Left y-axis: Output (black)



Silenced populations	Ouput correlation
X1 / R1 (T=0)	0.8576
X2 / R2 (T=0/1)	0.9834
X3 / R3 (T=0/1)	0.8785

FIGURE 5.22: **Side by side comparison of lesions in the minimal (left) and full (right) models.** $a = 1000$, $b = 1$. **A & E** No lesion. **B & F** X1-R1 lesioned. **C & G** X2-R2 lesioned. **D & H** X3-R3 lesioned. Bottom table: Correlation between output of reduced and reservoir model when one of the component is silenced. Right y-axis: Input (gray) Left y-axis: Output (black)

Chapter 6

Towards a synaptic working memory

In Chapter 5 we proposed a model which can maintain information in its neural activity. However tasks such as mental calculation require more than only the maintenance of information. They also require the inner manipulation of the information that is stored. In this Chapter we propose to extend our previous model with conceptors. We build conceptors that enforce some memory, and therefore represents that memory. By combining such conceptors, we show how it is possible to enforce an input-dependant behavior. These preliminary results suggest that conceptors can be used to manipulate the information that is stored.

COGNITIVE COMPUTATION

IN PRESS

Latent space exploration and functionalization of a gated working memory model using conceptors

Abstract: Introduction. Working memory is the ability to maintain and manipulate information. We introduce a method based on conceptors that allows us to manipulate information stored in the dynamics (latent space) of a gated working memory model. **Methods.** This latter model is based on a reservoir: a random recurrent network with trainable readouts. It is trained to hold a value in memory given an input stream when a gate signal is on and to maintain this information when the gate is off. The memorized information results in complex dynamics inside the reservoir that can be faithfully captured by a conceptor. **Results.** Such conceptors allow us to explicitly manipulate this information in order to perform various, but not arbitrary, operations. In this work, we show (1) how working memory can be stabilized or discretized using such conceptors, (2) how such conceptors can be linearly combined to form new memories, and (3) how these conceptors can be extended to a functional role. **Conclusion.** These preliminary results suggest that conceptors can be used to manipulate the latent space of the working memory even though several results we introduce are not as intuitive as one would expect.

Re-editing from the paper in press (Strock, Rougier, and Hinaut, 2020)

6.1 Introduction

A recent and major enhancement of the Reservoir Computing (RC) paradigm has been proposed by Jaeger (2014) and Jaeger (2017b) under the form of *conceptors* which are able to capture the subspace of internal states of a Recurrent Neural Network (RNN). In the case of Echo State Networks (ESN), conceptors can be used to capture the trajectory of reservoir states when the reservoir is fed with a particular input pattern (see figure 6.1). These conceptors allow to extend the capacity of the original ESNs by taking advantage of these new representations. For example, (Jaeger, 2014; Bao et al., 2016; Bartlett et al., 2019; Gast et al., 2017) showed how to use them for the recognition of temporal sequences while using them for the storage and retrieval of multiple temporal sequences. More recently, Mossakowski, Diaconescu, and Glauer (2019) proposed an implementation of fuzzy logic based on conceptors, while Liu, Ungar, and Sedoc (2019) used conceptors for online learning of sentence representations, and He and Jaeger (2018) proposed a general way to use conceptors during the learning of multiple tasks.

Conceptors yield several advantages when compared to classical reservoirs since Jaeger (2014) demonstrated that conceptors can be used for performing symbolic operations in the latent space of the different input patterns. Such symbolic operations have been already exploited in the framework of deep learning community and they provide impressive results. For instance, in natural language processing (NLP), Mikolov et al. (2013) showed that arithmetic operations such as "$king - men + woman$" give a vector similar to "*queen*". More recently, Brock et al. (2016) proposed a method to edit global image features based on operations performed on the latent space of generative adversarial networks (GANs). Conceptors provide similar logical operations in the framework of the reservoir computing paradigm. For instance, Jaeger (2014) proposes an operator that quantifies if a stimulus is similar to an already known conceptor. By associating one conceptor per class, it is possible to measure if a stimulus belongs to a class (positive evidence) or none (negative evidence). Beyond logical operations, linear combinations of conceptors allow to implement continuous morphing between set of states: they were used to create morphing between two time series corresponding to the extended interpolation of the time series (e.g. a morphing between two sine-waves with different frequencies is a sine-wave with an intermediate frequency).

This capacity of performing operations in the latent space resonates strongly with the notion of working memory (WM) as found in neuroscience. It is generally defined as the capacity to hold information for a short period of time as well as the capacity to manipulate this information in order to achieve some task or to reach a specific goal. In this context, we have introduced in (Strock, Hinaut, and Rougier, 2020) a reservoir with feedback connections that implements a gated working memory, i.e. a generic mechanism to maintain information at a given time (corresponding to when the gate is on, see figure 6.3). In this model, the memory is encoded in the dynamics of the

reservoir and information can be maintained without any sustained activity. This absence of sustained activity is precisely what makes it difficult to manipulate the underlying information and this is also the reason why some authors (Mongillo, Barak, and Tsodyks, 2008b; Stokes, 2015b; Masse et al., 2019b; Manohar et al., 2019) have suggested the existence of a mechanism to temporarily store information in synaptic weights. In this context, conceptors provide a plausible explanation for such a transfer as well as an explicit method for manipulating information; even if the conceptor mechanisms are for the moment not as biologically plausible as the reservoirs themselves.

In this article, we explore the nature of operations carried on by such conceptors and explore the different ways to combine them such as to explicitly manipulate memories. Even though the results we introduce in this article are preliminary and to some extents, counter-intuitive, this leads us to consider the notion of functional conceptors that would allow to arbitrarily manipulate working memory in the latent space.

6.2 Methods

6.2.1 Conceptors overview

Considering an ESN R that has been trained[1] to produce the sequence O_1 when presented with input sequence I_1, Jaeger (2014) demonstrated that it is possible to build an ESN $R*$ that will spontaneously produce the sequence O_1, in the absence of any input (see Figure 6.1). The activity of this new $R*$ can be decoded using the read-out weights of R. This is actually similar to the principle of the full-FORCE method introduced in (DePasquale et al., 2018) where internal weights are trained to match the internal activity of a teacher network receiving the desired output as input. However, Jaeger (2014) principle is applicable to multiple input patterns with the assumption that each input pattern makes the reservoir evolve in a separable region of the internal high dimensional space. In order to build $R*$, he proposed to approximate the activity of R when it receives any of these input patterns. Then he equips $R*$ with a set of recurrent connections (i.e. the conceptors) that are specific to each couple of input/output and that will project the internal state into the relevant sub-space. Jaeger (2017b) shows in particular how these conceptors can be considered as long-term memories for temporal patterns. Conceptors can store temporal patterns and reactivate them later with negligible loss of recall/precision. More generally, conceptors can be considered as long-term memories of internal states subspaces.

[1] offline with ridge regression

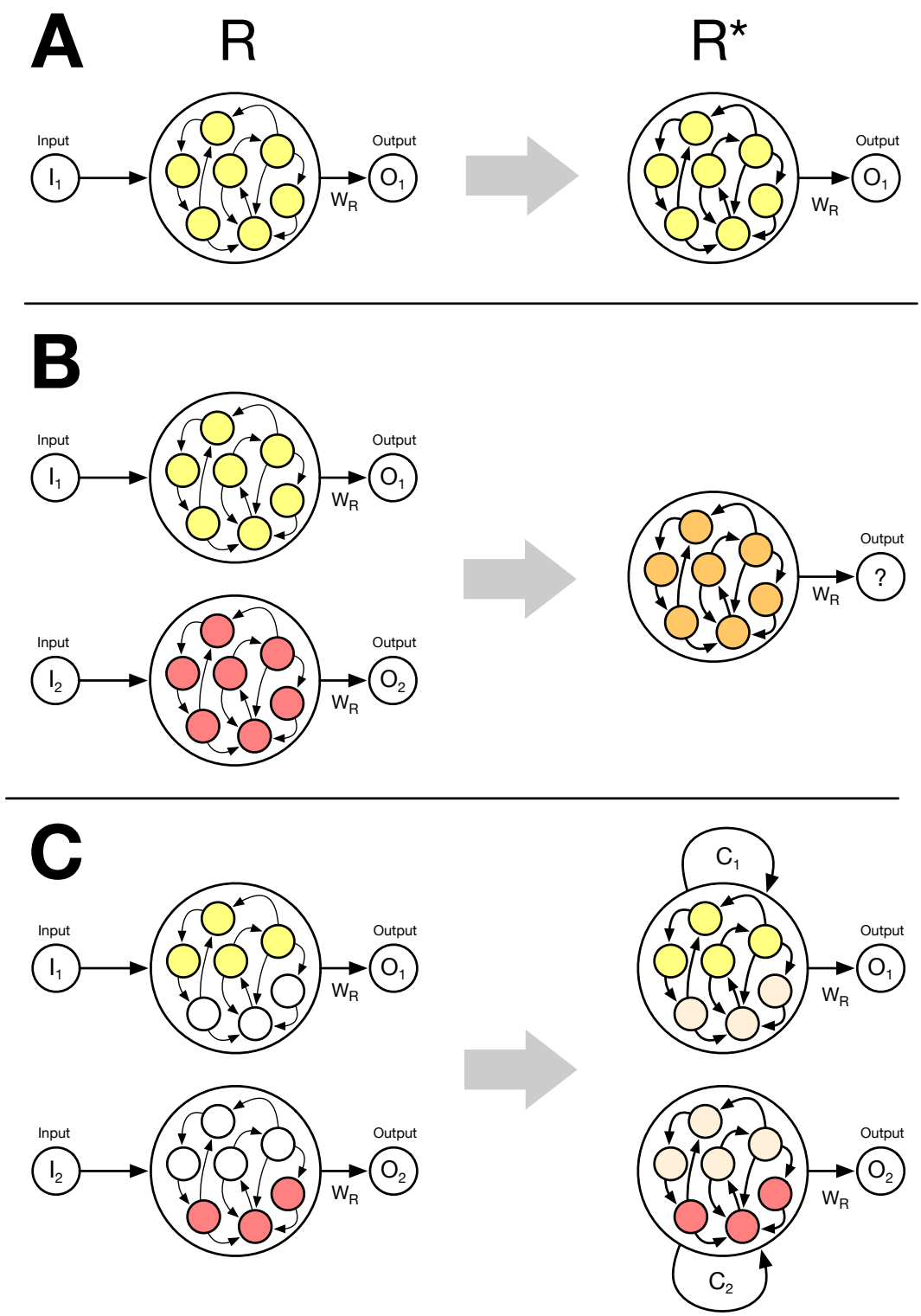


FIGURE 6.1: **Conceptors general idea A** Considering an ESN R that outputs the sequence O_1 when an input I_1 is presented, it is possible to build an ESN $R*$ that spontaneously outputs the sequence O_1 in the absence of any input. **B** Considering an ESN R that respectively outputs the sequences O_1 and O_2 when input I_1 and I_2 are presented, it is not possible to define $R*$ as in **A** since we cannot define the expected output in the absence of input. **C** Considering an ESN R that respectively outputs the sequences O_1 and O_2 when input I_1 and I_2 are presented – with the supplementary conditions that the inner representation corresponding to inputs I_1 and I_2 are separable – it is possible to build an ESN $R*$ equipped with a set of feedback weights C_1 or C_2 such that using C_1, $R*$ spontaneously outputs O_1 and using C_2, $R*$ spontaneously outputs O_2.

6.2.2 Models

Echo State Networks (ESN)

In this work we consider Echo State Networks (ESN) with feedback from readout units to the reservoir (Jaeger, 2001b). The system is described by the following update equations:

$$\begin{aligned} x[n] &= \tanh\left(W_{in}u[n] + Wx[n-1] + W_{fb}y[n-1]\right) + \xi \\ y[n] &= W_{out}x[n] \end{aligned}$$

where $u[n]$, $x[n]$ and $y[n]$ are respectively the input, the reservoir and the output at time n. W, W_{in}, W_{fb} and W_{out} are respectively the recurrent, the input, the feedback, the output weight matrices and ξ is a uniform white noise term added to reservoir units.

Controlling ESN dynamics using a conceptor

Following (Jaeger, 2014) notations, the equation for a conceptor C enforcing some particular dynamics can be written as:

$$x[n] = C\tanh\left(Wx[n-1] + b\right)$$

where C is the conceptor (possibly changing over time), $x[n]$ is the state of the model at time n, W is the recurrent matrix and b is a constant bias. This can be extended to the general case where we also have an input $u[n]$ (with input matrix W_{in}) (or similarly a feedback), and writes:

$$\begin{aligned} x[n] &= C\tanh\left(Wx[n-1] + W_{in}u[n] + W_{fb}y[n-1]\right) \\ y[n] &= W_{out}x[n] \end{aligned}$$

Using a conceptor C is similar to a change of W in $\tilde{W} = WC$ (and W_{out} in $W_{\text{out}}C$ if there is feedback). In our implementation, we thus consider:

$$\begin{aligned} x[n] &= \tanh\left(\left(W + W_{fb}W_{out}\right)C\,x[n-1] + W_{in}u[n]\right) \\ y[n] &= W_{out}x[n] \end{aligned}$$

Computing conceptors

In order to compute a conceptor for some given dynamics, it is necessary to collect all the states of the reservoir and to concatenate them in a matrix X. The conceptor C is then defined as:

$$C = XX^T\left(XX^T + \alpha^{-2}I\right)^{-1} = R\left(R + \alpha^{-2}I\right)^{-1}$$

where $R = XX^T$ is similar to a covariance matrix, and α (a.k.a the aperture) controls how close from the identity matrix C is.

Aperture adaptation

Intuitively, the aperture of a conceptor controls the precision of the internal states representation. However, no information on internal states is lost, because it is possible to change the aperture of a conceptor C without the need to recompute the conceptor from scratch. To change the aperture, one only need to adapt the conceptor C as follows:

$$\phi(C, \gamma) = C \left(C + \gamma^{-2}(I - C) \right)^{-1}$$

where $\phi(C, \gamma)$ represents the same states than C with a different aperture, and γ is controlling how the aperture is modified. Intuitively, $\phi(C, \gamma)$ modifies the aperture of C by a factor of γ.

Linear combination

Given two conceptors C and B and $\lambda \in \mathbb{R}$, the linear combination of conceptor C and B is defined as:

$$C = \lambda C + (1 - \lambda)B$$

In the following when $\lambda \in [0, 1]$ we will talk about interpolation, when $\lambda > 1$ about right-extrapolation, and when $\lambda < 0$ about left-extrapolation.

Boolean operations

Boolean operations can be written as:

$$\begin{aligned} C \vee B &= \left(I + \left(C(I - C)^{-1} + B(I - B)^{-1} \right)^{-1} \right)^{-1} \\ C \wedge B &= \left(C^{-1} + B^{-1} - I \right)^{-1} \\ \neg C &= I - C \end{aligned}$$

However, as highlighted in (Mossakowski, Diaconescu, and Glauer, 2019), $\vee$ and $\wedge$ are not idempotent (i.e. $C \vee C \neq C$ and $C \wedge C \neq C$). More precisely if C (resp. B) is a conceptor built with the covariance matrix R (resp. Q), Jaeger proposes to build $C \vee B$ using the covariance matrix $R + Q$ that is by design not idempotent. What we propose here is to consider instead the matrix $\beta R + (1 - \beta)Q$ with $\beta \in [0, 1]$ instead of $R + Q$, or if we want it to be symmetric $(R + Q)/2$. Similar calculation gives the following new $\vee_\beta$ and $\wedge_\beta$.

$$\begin{aligned} C \vee_\beta B &= \left(I + \left(\beta C(I - C)^{-1} + (1 - \beta)B(I - B)^{-1} \right)^{-1} \right)^{-1} \\ C \wedge_\beta B &= \left(\beta C^{-1} + (1 - \beta)B^{-1} \right)^{-1} \end{aligned}$$

This way of building the OR operation also has a data driven intuition. If we note $\beta = \frac{n}{n+p}$ where n (resp. p) is the number of data points used to build R (resp. Q) then $bR + (1-b)Q$ is the "correlation matrix" obtained by taking the union of all the data points. Moreover, if we choose $\beta = 0.5$ then there is a direct link between the two ways of defining the OR operator: $C \vee B = \phi(C \vee_{0.5} B, 2)$. In this study, the aperture was mostly not influencing the results, thus we show only the results for $\vee$.

6.2.3 Gating task

We consider the gating task described in (Strock, Hinaut, and Rougier, 2020). In this task the model receives an input V that is continuously varying over time and another input being either 0 or 1 (trigger or gate T). To complete the task, the output has to be updated to the value of the input when the trigger is active and to remain constant otherwise (similarly to a line attractor). In other words, the trigger acts as a gate that controls the entry of the value in the memory (the output). Figure 6.2 describes this task.

FIGURE 6.2: Gating task. Each column represents a time step (time increases from left to right), colored discs represent inputs (V and T) and the output (M).

Gated Working Memory Reservoir

We consider a reservoir with feedback from readout units to the reservoir (see Figure 6.3). In (Strock, Hinaut, and Rougier, 2020) we showed that this gated working memory reservoir is able to learn to robustly gate information in presence of noise and of a distracting input. The model behaves as a *line attractor* even if few values are used for training (about 10 values is enough). We also provided a minimal model version and showed an equivalence with GRU (Gated Recurrent Unit) cells (Cho et al., 2014a), which are a simplified version of Long-Short Term Memory (LSTM) cells.

6.2.4 Implementation details

We consider a reservoir of 1000 neurons that has been trained to solve a gating task described in Figure 5.2 **A**. The overall dynamics of the network we consider are described by the following equations:

$$x[n] = \tanh\left((W + W_{fb}W_{out})\ C\ x[n-1] + W_{in}u[n]\right) + \xi$$
$$y[n] = W_{out}x[n]$$

where $u[n]$, $x[n]$ and $y[n]$ are respectively the input, the reservoir and the output at time n. W, W_{in}, W_{fb}, W_{out} and C are respectively the recurrent,

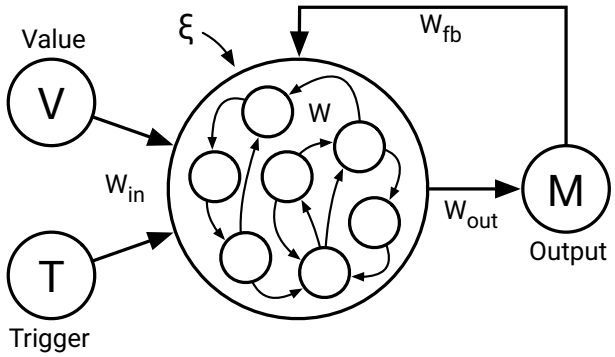


FIGURE 6.3: **The Gated Working Memory Reservoir model** The reservoir receives a random signal V in $[-1, +1]$ and a trigger signal T in $\{0, 1\}$. The output M is fed back to the reservoir.

the input, the feedback, the output and the conceptor weight matrices and ξ is a uniform white noise term added to reservoir units. W, W_{in}, W_{fb} are uniformly sampled between -1 and 1, and left untrained. Only W is scaled to have sparsity level equal to 0.5 and a spectral radius of 0.1. If not stated otherwise, the noise is selected uniformly between -10^{-4} and 10^{-4}.

The major difference with Jaeger's proposal in the way the patterns are stored is that in our case patterns are stored implicitly when solving the gating task. In other words, the patterns are stored by training W_{out} and not by explicitly recomputing an internal weight matrix. However as mentioned in (DePasquale et al., 2018), training W_{out} when there is a feedback is equivalent to recomputing the internal weight matrix W as $W^* = W + W_{fb}W_{out}$.

When W_{out} is computed to solve the gating task, the conceptor C is considered to be fixed and equal to the identity matrix ($C = I$). W_{out} is trained for 25,000 time steps. At each time step there is a 0.01 probability of having a trigger and the input value (V) is uniformly randomly sampled between -1 and 1. During training, the average holding time of the value in memory is therefore about 100 time steps.

After training, in normal mode, the conceptor C is equal to a conceptor C_m that is generated and associated to a constant value m. In order to compute this conceptor C_m, we impose a trigger ($T = 1$) as well as an input value ($V = m$) at the first time step, such that the reservoir has to maintain this value for 100 time steps. During these 100 time steps, we use the identity matrix in place of the conceptor. The conceptor C_m is then computed according to $C_m = XX^T \left(XX^T + \frac{I}{a}\right)^{-1}$, where X corresponds to the concatenation of all the 100 reservoir states following the trigger, each row corresponding to a time step, I the identity matrix and a the aperture. In all the experiments the aperture has been fixed to $a = 10$. For the conceptors pre-computed in Figure 6.4 and 6.6, the reservoir have been initialised with its last training state.

6.3 Results

6.3.1 Constant-memory conceptors

The idea behind what we named *constant-memory conceptors* is to capture explicitly the dynamics of a reservoir maintaining a value and to use this conceptor to later constrain the dynamics of any reservoir, inducing an alternative memory in the output as represented on figure 6.4. In order to build a constant-memory conceptor C_m, we consider the gated reservoir working memory model that receives a trigger and an input value m at time $t = 0$. We collect the states of the reservoir for 100 iterations from which we build the conceptor C_m and apply it immediately to the model. Results of this procedure is shown on figure 6.4B where five conceptors are built (gray bands) and applied immediately (white bands) without noticeable modification on the output of the reservoir since the actual dynamics and the dynamics stored in the conceptor are congruent. On figure 6.4A, we used the same procedure to build a set of 11 conceptors whose captured dynamics correspond to 11 values uniformly spread between -1 and +1. On figure 6.4C, after 100 iterations following the presentation of a new value, we apply the closest conceptor (Frobenius norm, inducing a distance between conceptors) from the previous set of 11 constant-memory conceptors. One can see that on figure 6.4C that the input value is first maintained (grey band) and jumps rapidly to the closest discretized value when the conceptor is actually applied and constrained the dynamics.

These constant-memory conceptors also exhibit a nice property regarding the long term maintenance of a memory. The initial gated working memory model is already quite robust regarding the long term maintenance of memory in the absence of internal noise (i.e. inside the reservoir). When it is trained for a few hundreds of time steps, it is able to maintain a memory for several thousands of time steps (see figure 6.5A) before the memory starts to slowly degrade. When conceptors are applied, this slow degradation vanishes (figure 6.5B): the RMSE without conceptor is of 4.21e-02 $\pm$ 1.84e-02 (mean $\pm$ std) whereas with conceptors it is 1.02e-03 $\pm$ 6.95e-04. In the presence of internal noise (inside the reservoir), the initial gated working memory model is much less robust and memory starts to degrade after only a few thousand time steps (see figure 6.5C). More precisely, a 10^{-4} noise ($std(\xi) = 10^{-4}$) prevents the model to maintain a value for a much longer time than the time that has been used to train it (figure 6.5C). However, when conceptors are applied, their benefit is even more obvious: after one hundred thousand time steps, without conceptor the memory converge towards a few values, whereas with conceptors the memory remains(figure 6.5D). The RMSE without conceptor becomes 1.39e-01 $\pm$ 7.66e-02 (3.3 times greater than without noise), whereas the RMSE with conceptors becomes 2.85e-03 $\pm$ 1.98e-03 (2.8 times greater than without noise).

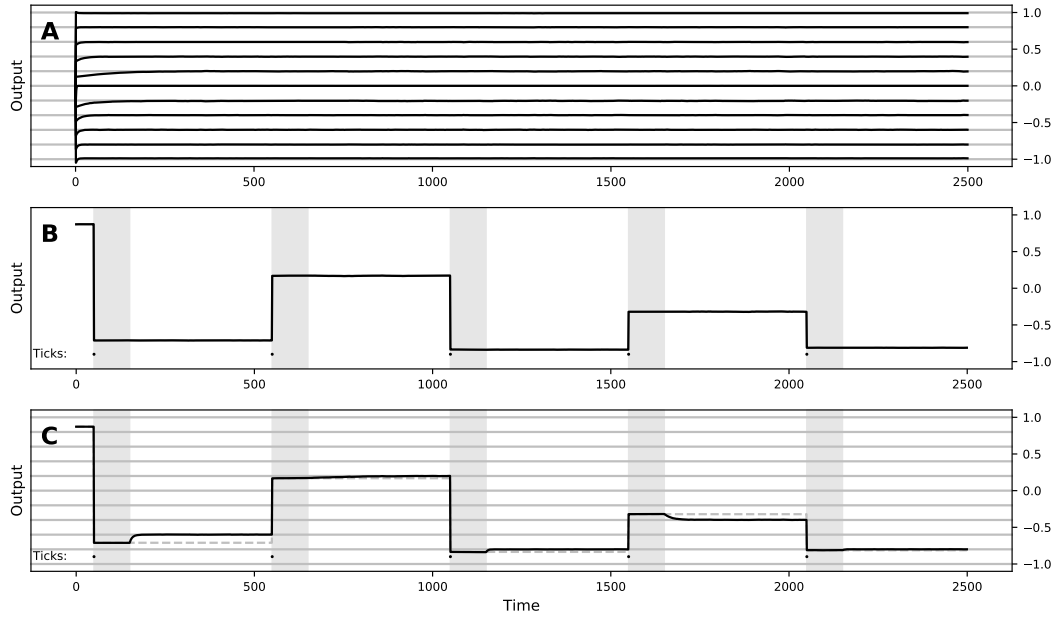


FIGURE 6.4: Approximation with conceptors and discrete conceptors. In **A**, **B** and **C** the model receives the same inputs across time. Black lines: Evolution of the reservoir readout y. Each black line in **A.** represents a different trial where a different conceptor is applied. Gray lines: the discrete value considered for each conceptor. Light gray areas: time period when conceptors are computed for the current value. **A.** Different trials showing various discrete conceptors applied. **B-C.** Conceptors C_m are computed using the 100 time steps following a trigger while $C = I$ (light gray areas). **B.** The conceptor C_m is directly applied during the 400 following time steps. **C.** The closest conceptor among the discretized conceptor is applied during the 400 following time steps. Dashed lines represents the memory that should have been kept if not discretized.

6.3.2 Linear interpolation of constant-memory conceptors

In Figure 6.6, we show two main ideas: **(1)** how a linear interpolation between two conceptors can allow to generalize the gating of other values, and **(2)** a representation of the space in which lies the conceptors and their link to the memory they encode. **(1)** Interpolation and extrapolation C of conceptor $C_{0.1}$ and conceptor $C_{1.0}$ has been computed as $C = \lambda C_{1.0} + (1 - \lambda)C_{0.1}$ with 31 λ values uniformly spread between -1 and 2. Even though the interpolated ($\lambda \in [0, 1]$) conceptors obtained are not exactly equivalent to C_m conceptors obtained in Figure 6.4, they seem to also correspond to a retrieved memory value that is maintained. The mapping between λ and the value is non-linearly encoded. For right-extrapolation ($\lambda \in [1, 2]$) the conceptor seems to be linked to a noisy version of a C_m conceptor: the output activity is not constant, but its moving average is constant. For left-extrapolation ($\lambda \in [-1, 0]$), the conceptor obtained does not seem to encode any information anymore: all the output activities collapse to zero. **(2)** Principal Component Analysis (PCA) have been performed using 201 pre-computed conceptors associated to values uniformly spread between -1 and 1. The first three components already explain approximately 85% of the variance. The first

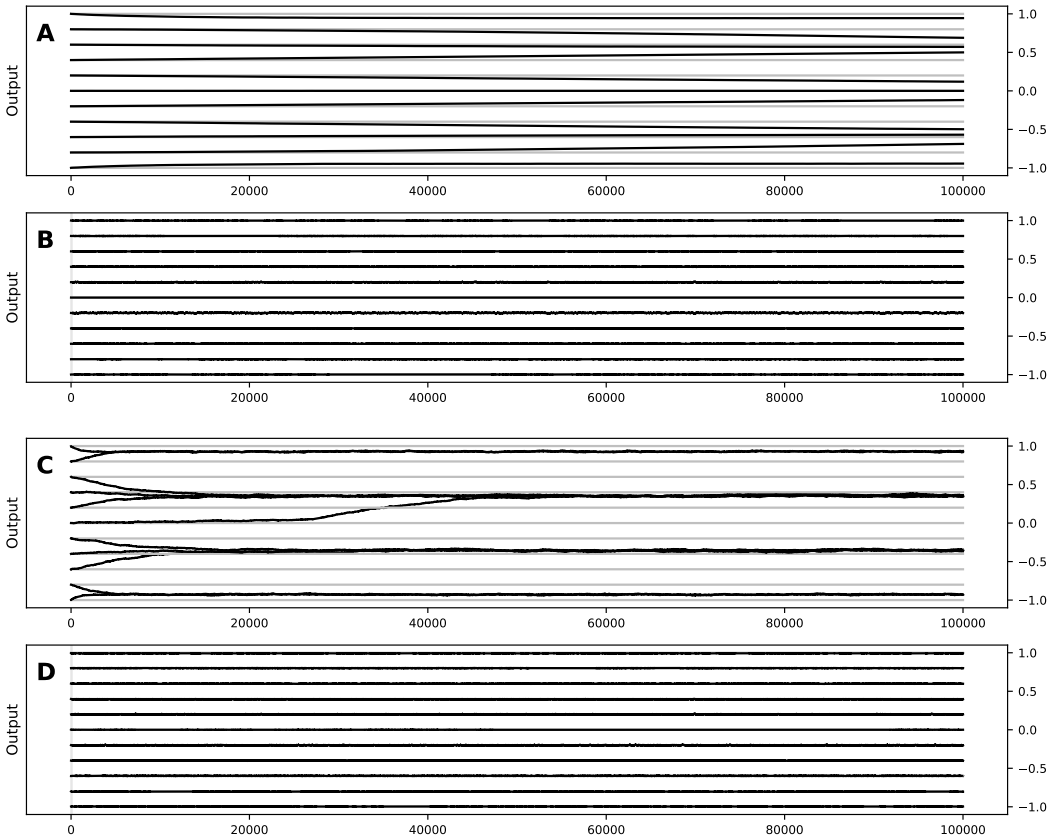


FIGURE 6.5: Stability comparison with or without conceptor, with or without noise. Black: Evolution of the readout y. lines: the discrete value considered. Light areas: time when conceptors are computed for the current value. **A**-**B** No noise. **C**-**D** 0.0001 noise. **A** and **C** No conceptor used. **B** and **D** Discrete conceptor are applied.

component seems to non-linearly encode the absolute value of the memory (Figure 6.6B) whereas the second component seems to non-linearly encode the memory itself (Figure 6.6C). The curved line composed of conceptors C_m (Figure 6.6E-G) shows visually why the extrapolation does not work as we expected.

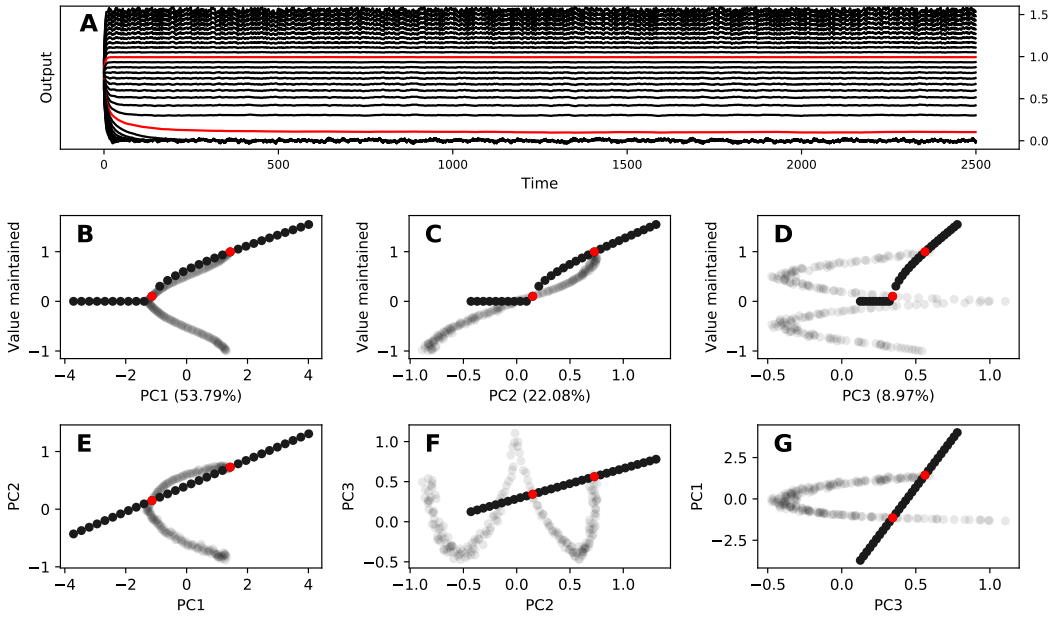


FIGURE 6.6: Generalisation of constant-memory conceptors C_m. **Red**: two constant-memory conceptors: $C_{0.1}$ and $C_{1.0}$. **Black**: Inferred conceptors, i.e. linear interpolation and extrapolation between $C_{0.1}$ and $C_{1.0}$. **A** Temporal evolution of the readout for different conceptors. **B-G** : constant-memory conceptors C_m for 201 values of m uniformly spread between -1 and 1. **B-D** Link between principal components of the conceptors and the memory they are encoding. For the interpolated conceptors, the memory is considered as the mean in the last 1000 time steps. **E-G** Representation of the conceptors in the three principal components of the C_m conceptors.

6.3.3 Functional conceptors

In this section, we show three examples where conceptors have a functional role: (1) constant-memory conceptors when triggers are received, (2) a conceptor enforcing triggers, and (3) the conjunction/disjunction of constant-memory conceptors. By *functional* we mean that conceptors do not only store and reactivate a dynamical pattern. Conceptors can do much more than constrain the dynamics in an attractor. Conceptors can also constrain the dynamics in a space where the behavior is input-dependent, i.e. the outputs are some function of the inputs.

Constant-memory conceptors when triggers are received. When a constant-memory conceptor C_m is applied to the gated working memory model, it seems to

quickly force the output y to match the value m, making it insensitive to the input value. However, the actual behavior is a bit different as illustrated on figure 6.7. On this figure, we can observe that the readout value of the resevoir under the influence of a conceptor $C_{0.5}$ is destabilized when a trigger occurs. More precisely, the trigger induce an instanteneous readout equal to the input until quickly relaxing to the constant value of the conceptor (0.5). This confirms that the combination of a reservoir and a constant-memory conceptor remains sensitive to the input. Said differently, a constant-memory conceptor C_m is not the mere storage of the value m but rather a function that constrains the dynamics of the reservoir such that when applied, the readout will eventually read m after some time.

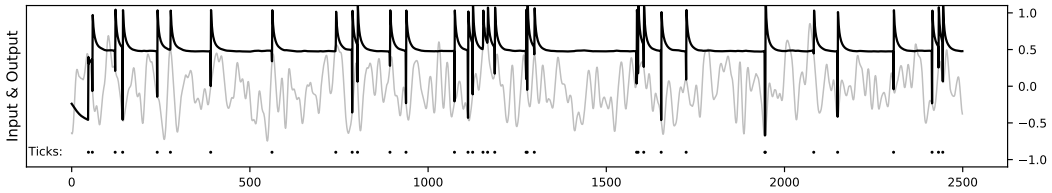


FIGURE 6.7: **Influence of trigger with constant-memory conceptors**. A constant-memory conceptor $C_{0.5}$ is applied while receiving several triggers. Gray: the input value (V). Black: Evolution of the readout y. When a trigger occurs (indicated by dots on the *Ticks* line) the readout transitorily goes to the current input value before going back to the value memorized by the conceptor.

Conceptor enforcing triggers. This functional aspect of conceptor can be further illustrated by building the following conceptor C_T: during 100 time steps, the reservoir receives constant triggers (T = 1) and has therefore to follow the value (V) it receives as input. Conceptor C_T is constructed from the states taken by the reservoir during these 100 time steps. Figure 6.8 shows what happens when this conceptor is subsequently applied: independently of the trigger signal, the output of the reservoir follows the input. Everything happens as if the reservoir was receiving a continuous trigger signal and the conceptor acts as a pass-through filter that modifies the behavior of the gated working memory as a whole (instead of modifying each individual value). This result suggests that conceptors may probably be extended to store arbitrary functions that act in the latent space of the reservoir. We do not know yet how to specify such arbitrary functions inside a conceptor, but the preliminary results we introduced are encouraging even though more theoretical work is needed.

Conjunction/disjunction of constant-memory conceptors. The last example where we show this functional aspect of conceptors is the disjunction of conceptors. If the conceptors C_m and $C_{m'}$ represent the subspaces when the value stored are respectively m and m', the disjunction of C_m and $C_{m'}$, i.e. $C_m \vee C_{m'}$ would represent the union (or the sum) of such subspaces. Applying $C_m \vee C_{m'}$ might therefore constrain the dynamics in one of these two subspaces, enforcing the memory (i.e. the output y) to become either m or m'. $C_m \vee C_{m'}$ could thus

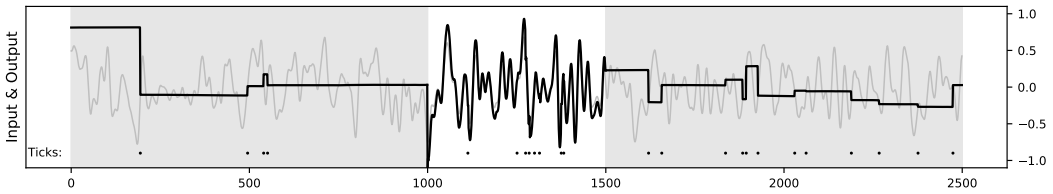


FIGURE 6.8: **Pass-through conceptors**. This conceptor is able to modify the behavior of the gated working model by letting all the values to pass through the reservoir up to the output, independently of the gating signal (time steps 1000 to 1500). Black: Evolution of the readout y. Gray: the input value (V). Light gray areas: no conceptor is applied (i.e. $C = I$). White areas: the conceptor simulating a constant trigger is applied (i.e. $C = C_T$), thus the input is redirected to the output.

store a choice function between m and m', or in other words, a sort of conditional assignment that would store m in some cases and m' in some others. It is not exactly what $C_m \vee C_{m'}$ does but it still implements some conditional assignment. When a trigger occurs the output jumps towards the value v to be maintained and then relaxes to another value that depends on v.

In Figure 6.9, we show the values towards which the output relaxes (i.e. relaxation values) when the disjunction of two constant-memory conceptors is applied. First, as the disjunction of twice the same conceptor $C_m \vee C_m$ is either equivalent to the same conceptor or to an aperture adaptation of it (i.e. $C_m \vee C_m = \phi(C_m, 2)$ and $C_m \vee_\beta C_m = C_m$), the value towards which the disjunction $C_m \vee C_m$ relaxes is the value of the conceptor itself (i.e. m). Then, we realized that we could predict what would be the relaxation values in different cases: in general the relaxation value was mostly either almost zero or the maximum of the absolute values of the two conceptors multiplied by the sign of the new value to be maintained. We propose the following formula to predict the value towards which $C_{m_1} \vee C_{m_2}$ relaxes:

$$v_{\text{relax}}(m_1, m_2, v) = \begin{cases} m_1 & \text{if } m_1 = m_2 \\ sign(v) \times max(|m_1|, |m_2|) & \text{if } min(|m_1|, |m_2|) < |v| \\ & \text{or } m_1 = -m_2 \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

where v is the initial value (V) proposed along with the trigger, m_1 (resp. m_2) is the constant associated to conceptor m_1 (resp. m_2), $v_{\text{relax}}(m_1, m_2, v)$ is the ultimate value reached while applying conceptor $C_{m_1} \vee C_{m_2}$. Said differently, the conceptor $C_{m_1} \vee C_{m_2}$ implements a conditional assignment (modulo the sign): if the input value at time of trigger is bigger than the minimum between m_1 and m_2 then it will converge towards the maximum of m_1 and m_2, otherwise it will converge towards 0. The predictions made by the formula are less accurate for extreme values such as for $v = 1.00$ (see Figure 6.9).

We hypothesize a similar formula for relaxation values of n constant-memory conceptors:

$$v_{\text{relax}}(m_1, ..., m_n, v) = \begin{cases} m_1 & \text{if } m_1 = ... = m_n \\ sign(v) \times max(|m_1|, ..., |m_n|) & \text{if } min(|m_1|, ..., |m_n|) < |v| \\ & \text{or } (\forall i, j \ |m_i| = |m_j| \\ & \text{and } \exists i, j \text{ such that } i > j \text{ and } m_i = -m_j \\ 0 & \text{otherwise} \end{cases}$$

where v is the initial value (V) given along with the trigger, m_i is the constant associated to conceptor C_{m_i}, $v_{\text{relax}}(m_1, m_2, ..., m_n, v)$ is the ultimate value reached while applying conceptor $\bigvee_{i=1}^{n} C_{m_i}$.

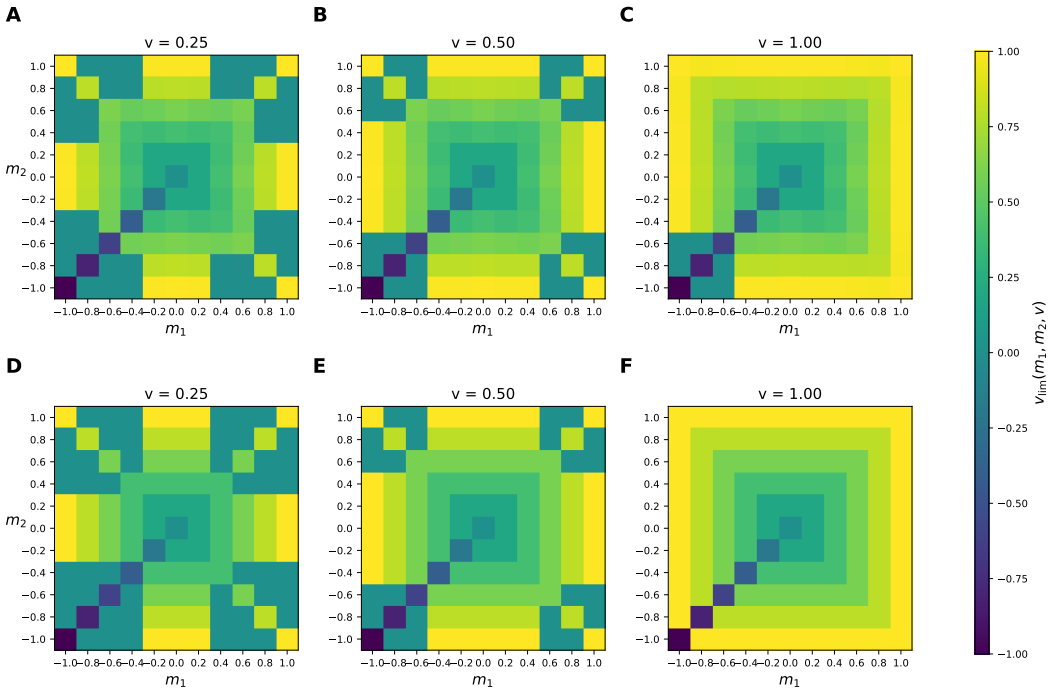


FIGURE 6.9: **Relaxation values** (i.e. values towards which the output converges) when applying the disjunction of two constant-memory conceptors. In other words, it corresponds to the final values reached when applying the conceptor $C_{m_1} \vee C_{m_2}$. **A-C** Empirical results from experiments. **D-F** Predictions based on equation 6.1.

We also studied how the conjunction constant-memory conceptors were influencing the dynamics. If the disjunction of conceptors is similar to a union (or sum), then the conjunction of conceptors is similar to an intersection. Moreover, as the memory is represented as the output, the memory cannot be simultaneously the value m and the value m' present. It is thus harder to predict what would be the behavior of such conceptor. In practice, as for the disjunction, when a trigger occurs the output jumps towards the value v to be maintained and then relaxes to another value. However, in that case the value towards which it relaxes is easy to describe, it is always almost zero. The conjunction of constant-memory conceptors acts therefore as C_0.

6.4 Discussion

Conceptors are powerful tools for performing explicit operations in the latent space of a reservoir even though the composition of such operations remains a difficult task. Using a reservoir model of gated working memory, we have shown another way to enforce a particular memory through the use of ad-hoc conceptors. These constant-memory conceptors therefore provide a synaptic form to the memory, and we have shown how they can be used to stabilize or discretize the memory. However, the effect of conceptors composition is counter-intuitive and largely differs from what we would naturally expect.

For instance, we have seen that the linear interpolation of constant-memory conceptors does not create another constant-memory conceptor, or at least it does not correspond to one we would have computed. The reason being that the space of constant-memory conceptors is not a straight line. Hence, they cannot be linearly interpolated: a mere linear combination of constant-memory conceptors could not lead to another constant-memory conceptor. Nevertheless, we have shown empirically that in all scenarios a linear combination of two constant-memory conceptors lead to a value that is maintained. However, this new memory is slightly oscillating around the combination of the constant values (see Figure 6.6). This oscillation being a direct consequence of the perturbation of the system (i.e. the input).

Moreover, the disjunction of constant-memory conceptors gives an example of functional conceptor. However, it does not implement what we expected. In the case of two conceptors with two constant values v_1 and v_2 such that $0 \leq v_1 \leq v_2$, we would expect that the disjunction encodes the two values simultaneously. More specifically, we expected such disjunction to implement a choice function (i.e. a conditional assignment) between the two values stored in each conceptor. Instead, the disjunction implements another conditional assignment, that does not converge towards v_1 but only towards 0 or v_2 depending on the given input value. To some extent, v_1 and v_2 influence the disjunction with different qualitative roles. Moreover, in the low rank case (i.e. when the recurrent weights are the sum of a low rank matrix and a random perturbation) only the largest fixed points can be stable (Schuessler et al., 2020b). Therefore, we can hypothesize that in the general case of a disjunction of $n > 2$ constant-memory conceptors, only the extreme value matter in the composite conceptor.

Even though our results are preliminary and will require more work to fully characterize how operations can be composed intuitively, this work opens the door to another form of working memory: a procedural (or functional) working memory. Instead of temporarily memorizing declarative information, this kind of working memory would be able to memorize procedural information (e.g. how a task should be performed, which processes should be applied, etc.). For instance, imagine you are given some instructions which are to sum up a series of numbers. In order to complete this task, it is necessary to keep track of the current sum (e.g. in a classical short-term declarative working memory) that needs to be updated each time a new number is given. However, it is also necessary to remember the preliminary instruction

(i.e summing up) in another form of working memory that needs to span the whole experiment and which is procedural in nature. This procedural nature makes this working memory quite peculiar because instead of memorizing a given information, it needs to memorize a procedure – here, a sequence of operations depending on the context – that needs to be applied each time an input is given. It is not yet clear how such memory could be encoded in the brain (e.g. sustained activity, dynamic activity, transient weights) and we think conceptors might be key in answering such a question, but more experimental and theoretical work will be needed. Similar conceptors than the ones we computed with our gated working memory reservoir model are likely to be computed with other working memory models (Lim and Goldman, 2013b; Nachstedt and Tetzlaff, 2017; Bouchacourt and Buschman, 2019b): it would be interesting to see whether the functional conceptors obtained are similar (i.e. our results would be generally applicable), or on the contrary, if differences occur.

Chapter 7

General Discussion

7.1 Summary of contributions

The first goal of this work (Chapter 5) was to study gated working memory and to design a robust neural architecture using a minimal set of hypotheses. More precisely, we did not make any assumptions regarding the actual mechanism (be it at the neural or circuit level) that would provide the "working memory property". Similarly, we did not make any assumptions regarding the internal activity of the system since we only considered if the information could or could not be decoded in the readout, rejecting the identification of neural activity and information. To do so, we positioned ourselves in the context of recurrent neural networks (RNNs) and designed a simple reservoir model with an input stream, a gating signal, and a readout that is fed-back to the reservoir. An extended sensitivity analysis showed that this model is actually very robust and works for a large range of parameters. Given our minimal set of hypotheses, this suggests that any neural population with random connections is able to implement such gated working memory. Furthermore, the model is highly resilient to noise and can be fed continuously with an input stream that severely impacts the dynamics of the reservoir. However, the logical gate we implemented allows for a very precise read-out of the memory when the gate is closed, making the model quasi insensitive to the input noise. This property has been illustrated through the design of a minimal model made of three neurons and whose equivalence with the full model has been demonstrated. In this minimal model, however, there exists a unit whose sustained activity is fully correlated with the gated working memory. Interestingly enough, such sustained activity units are not present in the full model and this suggests the activity is instead dynamically spread over the whole population of the reservoir. It is actually as if the memory would have been distributed over the whole population, making it resilient to neural damage. This model must be also considered with regards to other recent and similar models that all reject (as we do) the sustained activity hypothesis of working memory that has prevailed for a long time. This might deeply impact the field of neuroscience.

In the second part of this work (Chapter 6), and considering the distributed and dynamical nature of information in our gated working memory model, we have investigated how to explicitly manipulate the information that merely exists in the latent space of the reservoir. To do so, we extended the gated

working memory model using conceptors that allowed us to implement a synaptic-based working memory. Our initial idea was to use such conceptors in order to constrain the dynamics of the reservoir into specific subspaces, corresponding to the different states of the reservoir when a value is maintained. We thus investigated constant memory conceptors that are able to constrain the reservoir into the maintenance of a constant information, independently of the input value but for a much longer time compared to the initial model. Using several such conceptors, it is thus possible to build specific attractors towards which the initial stored value will converge. Said differently, after a short transitory state where the input value is memorized, the reservoir converge towards the nearest synaptically stored memory. This corresponds to a functional role that we can further take advantage of. More precisely, one of the nice features of the conceptors is that they can be combined into a new conceptor even though the behavior of the resulting conceptor might escape common sense. For example, we have seen that the linear combination of two constant conceptors creates a new conceptor that is not a constant-memory conceptor, but still maintains a slightly oscillating memory. Similarly, the disjunction of constant memory conceptors does not result in what we would intuitively expect, even though it brings interesting behavior such as a kind of conditional assignment. More work is needed to understand how to combine conceptors into controlled and useful operators. Our first intuition had been to combine using standard mathematical operators (linear combination, disjunction) but probably we need to consider new types of combinations such as to achieve a given behavior.

7.2 Limitations of the approach

Because we made so few structural hypotheses, it is hard to relate any of our neurons to any particular neurons in the brain. Thus, although our models can combine several visions of working memory, it is difficult to generate hypotheses that can be directly tested by experimentalists. However, we believe it is possible to extend the approach with more structure and derive similar results that might be easier to test experimentally. For instance, the reservoir could be formed by many regions of the brain (including for instance PFC), and the feedback signal that is essential to maintain in our model could be provided only to the regions which are assumed to maintain the information.

Moreover, it is not yet clear at an experimental level what generates the trigger signals, or how they are learned. In a model, O'Reilly proposes that these triggers could be generated by the Basal Ganglia that learns by trial and error to trigger at particular times in order to solve working memory tasks (O'Reilly and Frank, 2006a). But if PFC is in charge of attentional processes, PFC might be the one creating these triggers. In fact, these triggers might not need to be as explicit as they are in our model.

Furthermore, even though conceptor is a very useful theoretical concept, it is still hard to see how it could be biologically implemented. In our second study, we show that when a memory is enforced by a conceptor, it can still be changed by the trigger mechanism. However, this new memory won't last

long and is soon replaced by the memory enforced by the conceptor. That does not let in practice enough time to compute a new conceptor enforcing the new memory. In order to be able to compute the new conceptor, the applied conceptor need to be reset to a "no memory" conceptor (i.e. identity). If Jaeger shows how a conceptor can be computed online, there are not yet any proposed mechanisms on how to trigger a more abrupt change of conceptors, or even on how to combine online conceptors.

7.3 Perspectives

Using the reservoir computing paradigm, we have designed a simple recurrent neural architecture to implement a robust model of gated working memory. Interestingly enough and even though the set of hypothesis we have used is quite minimal, it can be actually further reduced using a simple rewriting rule of the inner weights. More precisely, the feedback from the output neuron to the reservoir can be subsumed after learning in the inner weights, therefore removing the feedback. In fact, this synaptic transfer could even be done through learningDePasquale et al. (2018). It may be that when feedback is needed, some neurons are recruited to give the feedback, and then the feedback would be absorbed. If such a mechanism is present in biological neurons, our model would provide another explanation of why there might be sustained activity in some cases and not in others. Moreover, we believe our model could be compared with an equivalent in-vitro culture of neurons on which we could test some of our hypotheses. Of course, this would require to translate first our rate-based model of neuron into a spiking model and to re-interpret our results. The main difficulty would be to learn and interpret the readout properly with regards to spike trains we may observe. However, such an experimental setup is beyond our expertise and cannot be done without cooperation with neurobiologists. Nonetheless, we think our model would help in reinterpreting the dynamics of such a neural population.

On the theoretical side, we have shown that the echo state property (ESP), i.e. fading memory property, might actually be a property too strong to ask for. When we require a memory to be extracted and maintained, possibly for a very long time, it is not possible to assume in the meantime that the reservoir enforces the echo state property. However, we have seen experimentally that even if the ESP is violated, it is violated in a few directions only, i.e. in the directions in which information is maintained, while other directions seem to satisfy the ESP. For instance, in the three-neurons model, only the neuron which maintains the memory does not satisfy the ESP. Similarly, in the reservoir when n memories are maintained, only n directions do not satisfy the ESP. It would be thus necessary to define and characterise a weaker property. Moreover, we have shown that this three-neurons model has a very similar behavior than the memory cell of a gated reccurent unit (GRU). It would be thus interesting to test if a network of such three-neurons model can solve tasks with a success level comparable as those of GRU. This would provide

an explanation for the real advantage of GRUs over SRNs: a well-chosen initialization of weights, and/or a well chosen set of weights to train.

Going back to our initial example regarding the neural mechanisms supporting simple mental calculations, we have partly answered the question by providing a mechanism to memorize a value as well as some mechanisms for explicitly manipulating memorized representations. We believe these latter mechanisms are close to a form of procedural memory that would allow to arbitrarily manipulate working memory. However, this would require more theoretical work, especially to explain how to enforce a given operation (e.g. addition, subtraction) in the latent space. We have started such theoretical work and have obtained encouraging, but very preliminary, results that are thus not introduced in this thesis. We think this last model would help us to understand what could be the neural mechanisms that allow for arithmetic operations to be carried out in the brain.

Bibliography

Abeles, Moshe (2012). *Local cortical circuits: an electrophysiological study*. Vol. 6. Springer Science & Business Media.

Aggleton, J.P., P.R. Hunt, and J.N.P. Rawlins (Feb. 1986). "The effects of hippocampal lesions upon spatial and non-spatial tests of working memory". In: *Behavioural Brain Research* 19.2, pp. 133–146. DOI: 10.1016/0166-4328(86)90011-2. URL: https://doi.org/10.1016/0166-4328(86)90011-2.

Amari, Shun ichi (1977). "Dynamics of pattern formation in lateral-inhibition type neural fields". In: *Biological Cybernetics* 27.2, pp. 77–87. DOI: 10.1007/bf00337259.

Atkinson, R.C. and R.M. Shiffrin (1968). "Human Memory: A Proposed System and its Control Processes". In: *Psychology of Learning and Motivation*. Elsevier, pp. 89–195. DOI: 10.1016/s0079-7421(08)60422-3. URL: https://doi.org/10.1016/s0079-7421(08)60422-3.

Baddeley, Alan (Nov. 2000). "The episodic buffer: a new component of working memory?" In: *Trends in Cognitive Sciences* 4.11, pp. 417–423. DOI: 10.1016/s1364-6613(00)01538-2. URL: https://doi.org/10.1016/s1364-6613(00)01538-2.

– (Jan. 2012). "Working Memory: Theories, Models, and Controversies". In: *Annual Review of Psychology* 63.1, pp. 1–29. DOI: 10.1146/annurev-psych-120710-100422. URL: https://doi.org/10.1146/annurev-psych-120710-100422.

Baddeley, Alan D and Graham Hitch (1974). "Working memory. The psychology of learning and motivation". In: *New York, NY: Academicp*.

Bancroft, Tyler D., William E. Hockley, and Philip Servos (2011). "Vibrotactile Working Memory as a Model Paradigm for Psychology, Neuroscience, and Computational Modeling". In: *Frontiers in Human Neuroscience* 5. DOI: 10.3389/fnhum.2011.00162. URL: https://doi.org/10.3389/fnhum.2011.00162.

Bao, Jiao et al. (2016). "Action recognition based on conceptors of skeleton joint trajectories". In: *Rev. Fac. Ing* 31.4, pp. 11–22.

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

Barak, Omri and Misha Tsodyks (Apr. 2014). "Working models of working memory". In: *Current Opinion in Neurobiology* 25, pp. 20–24. DOI: 10.1016/j.conb.2013.10.008.

Barak, Omri et al. (Apr. 2013). "From fixed points to chaos: Three models of delayed discrimination". In: *Progress in Neurobiology* 103, pp. 214–222. DOI: 10.1016/j.pneurobio.2013.02.002. URL: https://doi.org/10.1016/j.pneurobio.2013.02.002.

Barrouillet, Pierre (2018). "Numerical Cognition and Memory(ies)". In: *Heterogeneity of Function in Numerical Cognition*. Elsevier, pp. 361–385. DOI: 10.1016/b978-0-12-811529-9.00017-0. URL: https://doi.org/10.1016/b978-0-12-811529-9.00017-0.

Bartlett, Madeleine et al. (2019). *Recognizing Human Internal States: A Conceptor-Based Approach*. arXiv: 1909.04747 [cs.HC].

Bechara, Antoine et al. (Jan. 1998). "Dissociation Of Working Memory from Decision Making within the Human Prefrontal Cortex". In: *The Journal of Neuroscience* 18.1, pp. 428–437. DOI: 10.1523/jneurosci.18-01-00428.1998.

Beer, Chen and Omri Barak (2019). "One step back, two steps forward: interference and learning in recurrent neural networks". In: *arXiv preprint arXiv:1805.09603v2*.

Bengio, Y., P. Simard, and P. Frasconi (Mar. 1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166. DOI: 10.1109/72.279181. URL: https://doi.org/10.1109/72.279181.

Bouchacourt, Flora and Timothy J. Buschman (July 2019a). "A Flexible Model of Working Memory". In: *Neuron* 103.1, 147–160.e8. DOI: 10.1016/j.neuron.2019.04.020. URL: https://doi.org/10.1016/j.neuron.2019.04.020.

– (July 2019b). "A Flexible Model of Working Memory". In: *Neuron* 103.1, 147–160.e8. DOI: 10.1016/j.neuron.2019.04.020.

Brock, Andrew et al. (2016). "Neural photo editing with introspective adversarial networks". In: *arXiv preprint arXiv:1609.07093*.

Brown, John (Feb. 1958). "Some Tests of the Decay Theory of Immediate Memory". In: *Quarterly Journal of Experimental Psychology* 10.1, pp. 12–21. DOI: 10.1080/17470215808416249. URL: https://doi.org/10.1080/17470215808416249.

Buonomano, D V and M M Merzenich (1995). "Temporal information transformed into a spatial code by a neural network with realistic properties". In: *Science* 267.5200, pp. 1028–1030.

Camos, Valérie (2018). "Do Not Forget Memory to Understand Mathematical Cognition". In: *Heterogeneity of Function in Numerical Cognition*. Elsevier, pp. 433–447. DOI: 10.1016/b978-0-12-811529-9.00020-0. URL: https://doi.org/10.1016/b978-0-12-811529-9.00020-0.

Chafee, Matthew V. and Patricia S. Goldman-Rakic (Mar. 2000). "Inactivation of Parietal and Prefrontal Cortex Reveals Interdependence of Neural Activity During Memory-Guided Saccades". In: *Journal of Neurophysiology* 83.3, pp. 1550–1566. DOI: 10.1152/jn.2000.83.3.1550. URL: https://doi.org/10.1152/jn.2000.83.3.1550.

Chai, Wen Jia, Aini Ismafairus Abd Hamid, and Jafri Malin Abdullah (Mar. 2018). "Working Memory From the Psychological and Neurosciences Perspectives: A Review". In: *Frontiers in Psychology* 9. DOI: 10.3389/fpsyg.2018.00401. URL: https://doi.org/10.3389/fpsyg.2018.00401.

Cho, Kyunghyun et al. (Oct. 2014a). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1724–1734. URL: http://www.aclweb.org/anthology/D14-1179.

Cho, Kyunghyun et al. (2014b). *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. arXiv: 1406.1078 [cs.CL].

Christophel, Thomas B. et al. (Feb. 2017). "The Distributed Nature of Working Memory". In: *Trends in Cognitive Sciences* 21.2, pp. 111–124. DOI: 10.1016/j.tics.2016.12.007. URL: https://doi.org/10.1016/j.tics.2016.12.007.

Christopher Olah (2015). *Understanding LSTM Networks*. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Online; accessed 7-September-2020].

Compte, A. (Sept. 2000). "Synaptic Mechanisms and Network Dynamics Underlying Spatial Working Memory in a Cortical Network Model". In: *Cerebral Cortex* 10.9, pp. 910–923. DOI: 10.1093/cercor/10.9.910.

Compte, Albert (Apr. 2006). "Computational and in vitro studies of persistent activity: Edging towards cellular and synaptic mechanisms of working memory". In: *Neuroscience* 139.1, pp. 135–151. DOI: 10.1016/j.neuroscience.2005.06.011.

Conrad, R. and A. J. Hull (Nov. 1964). "INFORMATION, ACOUSTIC CONFUSION AND MEMORY SPAN". In: *British Journal of Psychology* 55.4, pp. 429–432. DOI: 10.1111/j.2044-8295.1964.tb00928.x. URL: https://doi.org/10.1111/j.2044-8295.1964.tb00928.x.

Constantinidis, Christos et al. (Aug. 2018a). "Persistent Spiking Activity Underlies Working Memory". In: *The Journal of Neuroscience* 38.32, pp. 7020–7028. DOI: 10.1523/jneurosci.2486-17.2018. URL: https://doi.org/10.1523/jneurosci.2486-17.2018.

– (Aug. 2018b). "Persistent Spiking Activity Underlies Working Memory". In: *The Journal of Neuroscience* 38.32, pp. 7020–7028. DOI: 10.1523/jneurosci.2486-17.2018.

Courtney, Susan M. et al. (Apr. 1997). "Transient and sustained activity in a distributed neural system for human working memory". In: *Nature* 386.6625, pp. 608–611. DOI: 10.1038/386608a0. URL: https://doi.org/10.1038/386608a0.

Cowan, Nelson (1999). "An embedded-processes model of working memory". In: *Models of working memory: Mechanisms of active maintenance and executive control* 20, p. 506.

– (2008). "Chapter 20 What are the differences between long-term, short-term, and working memory?" In: *Progress in Brain Research*. Elsevier, pp. 323–338. DOI: 10.1016/s0079-6123(07)00020-9. URL: https://doi.org/10.1016/s0079-6123(07)00020-9.

– (Feb. 2010). "The Magical Mystery Four". In: *Current Directions in Psychological Science* 19.1, pp. 51–57. DOI: 10.1177/0963721409359277. URL: https://doi.org/10.1177/0963721409359277.

– (Nov. 2016). "The many faces of working memory and short-term storage". In: *Psychonomic Bulletin & Review* 24.4, pp. 1158–1170. DOI: 10.3758/s13423-016-1191-6. URL: https://doi.org/10.3758/s13423-016-1191-6.

Craik, Fergus I.M. and Robert S. Lockhart (Dec. 1972). "Levels of processing: A framework for memory research". In: *Journal of Verbal Learning and Verbal Behavior* 11.6, pp. 671–684. DOI: 10.1016/s0022-5371(72)80001-x. URL: https://doi.org/10.1016/s0022-5371(72)80001-x.

Curry, Michaela D et al. (2017). "A Cage-Based Training System for Non-Human Primates". In: *AIMS Neuroscience* 4.3, pp. 102–119. DOI: 10.3934/neuroscience.2017.3.102. URL: https://doi.org/10.3934/neuroscience.2017.3.102.

Curtis, Clayton E. and Mark D'Esposito (Sept. 2003). "Persistent activity in the prefrontal cortex during working memory". In: *Trends in Cognitive Sciences* 7.9, pp. 415–423. DOI: 10.1016/s1364-6613(03)00197-9. URL: https://doi.org/10.1016/s1364-6613(03)00197-9.

Dambre, Joni et al. (July 2012a). "Information Processing Capacity of Dynamical Systems". In: *Scientific Reports* 2.1. DOI: 10.1038/srep00514. URL: https://doi.org/10.1038/srep00514.

– (July 2012b). "Information Processing Capacity of Dynamical Systems". In: *Scientific Reports* 2.1. DOI: 10.1038/srep00514.

Dehaene, Stanislas (1997). *La bosse des maths*. Odile Jacob.

DePasquale, Brian et al. (Feb. 2018). "full-FORCE: A target-based method for training recurrent networks". In: *PLOS ONE* 13.2. Ed. by Maurice J. Chacron, e0191527. DOI: 10.1371/journal.pone.0191527. URL: https://doi.org/10.1371/journal.pone.0191527.

D'Esposito, Mark and Bradley R. Postle (Jan. 2015). "The Cognitive Neuroscience of Working Memoryf". In: *Annual Review of Psychology* 66.1, pp. 115–142. DOI: 10.1146/annurev-psych-010814-015031. URL: https://doi.org/10.1146/annurev-psych-010814-015031.

Dominey, Peter F. (Aug. 1995). "Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning". In: *Biological Cybernetics* 73.3, pp. 265–274. DOI: 10.1007/bf00201428. URL: https://doi.org/10.1007/bf00201428.

Durstewitz, Daniel, Jeremy K. Seamans, and Terrence J. Sejnowski (Nov. 2000). "Neurocomputational models of working memory". In: *Nature Neuroscience* 3, pp. 1184–1191. DOI: 10.1038/81460.

Edin, F. et al. (Apr. 2009). "Mechanism for top-down control of working memory capacity". In: *Proceedings of the National Academy of Sciences* 106.16, pp. 6802–6807. DOI: 10.1073/pnas.0901894106.

Elman, Jeffrey L. (Mar. 1990). "Finding Structure in Time". In: *Cognitive Science* 14.2, pp. 179–211. DOI: 10.1207/s15516709cog1402_1. URL: https://doi.org/10.1207/s15516709cog1402_1.

Enel, Pierre et al. (June 2016a). "Reservoir Computing Properties of Neural Dynamics in Prefrontal Cortex". In: *PLOS Computational Biology* 12.6. Ed. by Jill X O'Reilly, e1004967. DOI: 10.1371/journal.pcbi.1004967. URL: https://doi.org/10.1371/journal.pcbi.1004967.

– (June 2016b). "Reservoir Computing Properties of Neural Dynamics in Prefrontal Cortex". In: *PLOS Computational Biology* 12.6. Ed. by Jill X O'Reilly, e1004967. DOI: 10.1371/journal.pcbi.1004967.

Funahashi, Shintaro (Apr. 2017a). "Working Memory in the Prefrontal Cortex". In: *Brain Sciences* 7.12, p. 49. DOI: 10.3390/brainsci7050049. URL: https://doi.org/10.3390/brainsci7050049.

– (Apr. 2017b). "Working Memory in the Prefrontal Cortex". In: *Brain Sciences* 7.12, p. 49. DOI: 10.3390/brainsci7050049.

Fuster, J. M. and G. E. Alexander (Aug. 1971). "Neuron Activity Related to Short-Term Memory". In: *Science* 173.3997, pp. 652–654. DOI: 10.1126/science.173.3997.652. URL: https://doi.org/10.1126/science.173.3997.652.

Fuster, Joaquin M. (May 2001). "The Prefrontal Cortex—An Update". In: *Neuron* 30.2, pp. 319–333. DOI: 10.1016/s0896-6273(01)00285-9.

Gallicchio, Claudio, Alessio Micheli, and Luca Pedrelli (2018a). "Comparison between DeepESNs and gated RNNs on multivariate time-series prediction". In: *CoRR* abs/1812.11527. arXiv: 1812.11527. URL: http://arxiv.org/abs/1812.11527.

– (Dec. 2018b). "Design of deep echo state networks". In: *Neural Networks* 108, pp. 33–47. DOI: 10.1016/j.neunet.2018.08.002. URL: https://doi.org/10.1016/j.neunet.2018.08.002.

Gandhi, Neeraj J. and Husam A. Katnani (July 2011). "Motor Functions of the Superior Colliculus". In: *Annual Review of Neuroscience* 34.1, pp. 205–231. DOI: 10.1146/annurev-neuro-061010-113728.

Gast, Richard et al. (Apr. 2017). "Encoding and Decoding Dynamic Sensory Signals with Recurrent Neural Networks: An Application of Conceptors to Birdsongs". In: *BioRxiv*. DOI: 10.1101/131052. URL: https://doi.org/10.1101/131052.

Gathercole, Susan E. (Nov. 1999). "Cognitive approaches to the development of short-term memory". In: *Trends in Cognitive Sciences* 3.11, pp. 410–419. DOI: 10.1016/s1364-6613(99)01388-1. URL: https://doi.org/10.1016/s1364-6613(99)01388-1.

Gers, F.A. and J. Schmidhuber (2000). "Recurrent nets that time and count". In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. IEEE. DOI: 10.1109/ijcnn.2000.861302. URL: https://doi.org/10.1109/ijcnn.2000.861302.

GoldmanRakic, Patricia S. (1987). "Circuitry of Primate Prefrontal Cortex and Regulation of Behavior by Representational Memory". In: *Comprehensive Physiology*. DOI: 10.1002/cphy.cp010509.

Graves, Alex (2013). *Generating Sequences With Recurrent Neural Networks*. arXiv: 1308.0850 [cs.NE].

Graves, Alex, Greg Wayne, and Ivo Danihelka (2014). *Neural Turing Machines*. arXiv: 1410.5401 [cs.NE].

Greff, Klaus et al. (Oct. 2017a). "LSTM: A Search Space Odyssey". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10, pp. 2222–2232. DOI: 10.1109/tnnls.2016.2582924. URL: https://doi.org/10.1109/tnnls.2016.2582924.

– (Oct. 2017b). "LSTM: A Search Space Odyssey". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10, pp. 2222–2232. DOI: 10.1109/tnnls.2016.2582924.

Grigoryeva, Lyudmila and Juan-Pablo Ortega (Dec. 2018). "Echo state networks are universal". In: *Neural Networks* 108, pp. 495–508. DOI: 10.1016/j.neunet.2018.08.025. URL: https://doi.org/10.1016/j.neunet.2018.08.025.

He, Xu and Herbert Jaeger (2018). "Overcoming Catastrophic Interference using Conceptor-Aided Backpropagation". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=B1al7jg0b.

Hebb, DO (1949). "The organization of behavior; a neuropsychological theory." In:

Hinaut, Xavier and Peter Ford Dominey (Feb. 2013). "Real-Time Parallel Processing of Grammatical Structure in the Fronto-Striatal System: A Recurrent Network Simulation Study Using Reservoir Computing". In: *PLoS ONE* 8.2. Ed. by Thomas Boraud, e52946. DOI: 10.1371/journal.pone.0052946.

Hinaut, Xavier et al. (Nov. 2015). "Corticostriatal response selection in sentence production: Insights from neural network simulation with reservoir computing". In: *Brain and Language* 150, pp. 54–68. DOI: 10.1016/j.bandl.2015.08.002.

Hochreiter, Sepp and Jürgen Schmidhuber (Nov. 1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

Hoerzer, G M, R Legenstein, and W Maass (2012). "Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning". In: *Cerebral cortex* 24.3, pp. 677–690.

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

Hunter, J. D. (2007). "Matplotlib: A 2D graphics environment". In: *Computing In Science & Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

Jaeger, Herbert (Jan. 2001a). *The "echo state" approach to analysing and training recurrent neural networks*. Tech. rep. 148. Bonn, Germany: German National Research Center for Information Technology GMD. URL: http://publica.fraunhofer.de/starweb/servlet.starweb?path=urn.web&search=urn:nbn:de:0011-b-731351.

Jaeger, Herbert (2001b). "The echo state approach to analysing and training recurrent neural networks-with an erratum note". In: *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148.34, p. 13.
– (2002). "Short term memory in echo state networks." In: *GMD - German National Research Institute for Computer Science*.
– (2007). "Echo state network". In: *Scholarpedia* 2.9, p. 2330. DOI: 10.4249/scholarpedia.2330. URL: https://doi.org/10.4249/scholarpedia.2330.
– (2012). *Long short-term memory in echo state networks: Details of a simulation study*. Tech. rep. Jacobs University Bremen.
– (2014). "Controlling recurrent neural networks by conceptors". In: *arXiv preprint arXiv:1403.3369*.
– (2017a). "Using Conceptors to Manage Neural Long-Term Memories for Temporal Patterns". In: *Journal of Machine Learning Research* 18.13, pp. 1–43. URL: http://jmlr.org/papers/v18/15-449.html.
– (2017b). "Using Conceptors to Manage Neural Long-Term Memories for Temporal Patterns". In: *Journal of Machine Learning Research* 18.13, pp. 1–43.
James, W (1890). *The principles of psychology, Vol. 2. Henry Holt and Company*.
Jing, Li et al. (2017). "Tunable efficient unitary neural networks (eunn) and their application to rnns". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pp. 1733–1741.
Jing, Li et al. (Apr. 2019). "Gated Orthogonal Recurrent Units: On Learning to Forget". In: *Neural Computation* 31.4, pp. 765–783. DOI: 10.1162/neco_a_01174. URL: https://doi.org/10.1162/neco_a_01174.
Jones, Eric, Travis Oliphant, and Pearu Peterson (2001). *SciPy: Open source scientific tools for Python*. URL: http://www.scipy.org.
Jonides, John et al. (July 1997). "Verbal Working Memory Load Affects Regional Brain Activation as Measured by PET". In: *Journal of Cognitive Neuroscience* 9.4, pp. 462–475. DOI: 10.1162/jocn.1997.9.4.462. URL: https://doi.org/10.1162/jocn.1997.9.4.462.
Jordan, Michael I. (1997). "Serial Order: A Parallel Distributed Processing Approach". In: *Neural-Network Models of Cognition - Biobehavioral Foundations*. Elsevier, pp. 471–495. DOI: 10.1016/s0166-4115(97)80111-2. URL: https://doi.org/10.1016/s0166-4115(97)80111-2.
Kim, Sung Soo et al. (May 2017). "Ring attractor dynamics in theDrosophilacentral brain". In: *Science* 356.6340, pp. 849–853. DOI: 10.1126/science.aal4835. URL: https://doi.org/10.1126/science.aal4835.
Koulakov, Alexei A. et al. (July 2002). "Model for a robust neural integrator". In: *Nature Neuroscience* 5.8, pp. 775–782. DOI: 10.1038/nn893.
Kubota, K and H Niki (May 1971). "Prefrontal cortical unit activity and delayed alternation performance in monkeys." In: *Journal of Neurophysiology* 34.3, pp. 337–347. DOI: 10.1152/jn.1971.34.3.337. URL: https://doi.org/10.1152/jn.1971.34.3.337.
LaRocque, Joshua J. et al. (Jan. 2013). "Decoding Attended Information in Short-term Memory: An EEG Study". In: *Journal of Cognitive Neuroscience* 25.1, pp. 127–142. DOI: 10.1162/jocn_a_00305. URL: https://doi.org/10.1162/jocn_a_00305.
Lashley, K. S. (1930). "Basic neural mechanisms in behavior." In: *Psychological Review* 37.1, pp. 1–24. DOI: 10.1037/h0074134. URL: https://doi.org/10.1037/h0074134.
Leavitt, Matthew L., Diego Mendoza-Halliday, and Julio C. Martinez-Trujillo (June 2017). "Sustained Activity Encoding Working Memories: Not Fully Distributed".

In: *Trends in Neurosciences* 40.6, pp. 328–346. DOI: 10.1016/j.tins.2017.04.004. URL: https://doi.org/10.1016/j.tins.2017.04.004.

Legenstein, Robert and Wolfgang Maass (Apr. 2007). "Edge of chaos and prediction of computational performance for neural circuit models". In: *Neural Networks* 20.3, pp. 323–334. DOI: 10.1016/j.neunet.2007.04.017.

Lewis-Peacock, Jarrod A. et al. (Jan. 2012). "Neural Evidence for a Distinction between Short-term Memory and the Focus of Attention". In: *Journal of Cognitive Neuroscience* 24.1, pp. 61–79. DOI: 10.1162/jocn_a_00140. URL: https://doi.org/10.1162/jocn_a_00140.

Lim, Sukbin and Mark S Goldman (Aug. 2013a). "Balanced cortical microcircuitry for maintaining information in working memory". In: *Nature Neuroscience* 16.9, pp. 1306–1314. DOI: 10.1038/nn.3492.

– (Aug. 2013b). "Balanced cortical microcircuitry for maintaining information in working memory". In: *Nature Neuroscience* 16.9, pp. 1306–1314. DOI: 10.1038/nn.3492.

Liu, Tianlin, Lyle Ungar, and João Sedoc (2019). "Continual Learning for Sentence Representations Using Conceptors". In: *CoRR* abs/1904.09187. arXiv: 1904.09187. URL: http://arxiv.org/abs/1904.09187.

Lukoeviius, Mantas (2012). "A Practical Guide to Applying Echo State Networks". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, pp. 659–686. DOI: 10.1007/978-3-642-35289-8_36.

Lundqvist, Mikael, Pawel Herman, and Earl K. Miller (Aug. 2018). "Working Memory: Delay Activity, Yes! Persistent Activity? Maybe Not". In: *The Journal of Neuroscience* 38.32, pp. 7013–7019. DOI: 10.1523/jneurosci.2485-17.2018. URL: https://doi.org/10.1523/jneurosci.2485-17.2018.

Maass, Wolfgang, Prashant Joshi, and Eduardo D. Sontag (2007). "Computational Aspects of Feedback in Neural Circuits". In: *PLoS Computational Biology* 3.1, e165. DOI: 10.1371/journal.pcbi.0020165.

Maass, Wolfgang, Thomas Natschläger, and Henry Markram (Nov. 2002). "Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations". In: *Neural Computation* 14.11, pp. 2531–2560. DOI: 10.1162/089976602760407955.

Machens, C. K., R. Romo, and C. D. Brody (Jan. 2010). "Functional, But Not Anatomical, Separation of "What" and "When" in Prefrontal Cortex". In: *Journal of Neuroscience* 30.1, pp. 350–360. DOI: 10.1523/jneurosci.3276-09.2010.

Madigna, Stephen A. (1971). "Modality and recall order interactions in short-term memory for serial order." In: *Journal of Experimental Psychology* 87.2, pp. 294–296. DOI: 10.1037/h0030549. URL: https://doi.org/10.1037/h0030549.

Mannella, Francesco and Gianluca Baldassarre (Nov. 2015). "Selection of cortical dynamics for motor behaviour by the basal ganglia". In: *Biological Cybernetics* 109.6, pp. 575–595. DOI: 10.1007/s00422-015-0662-6.

Manohar, Sanjay G. et al. (June 2019). "Neural mechanisms of attending to items in working memory". In: *Neuroscience & Biobehavioral Reviews* 101, pp. 1–12. DOI: 10.1016/j.neubiorev.2019.03.017.

Mante, Valerio et al. (Nov. 2013). "Context-dependent computation by recurrent dynamics in prefrontal cortex". In: *Nature* 503.7474, pp. 78–84. DOI: 10.1038/nature12742. URL: https://doi.org/10.1038/nature12742.

Masse, Nicolas Y. et al. (June 2019a). "Circuit mechanisms for the maintenance and manipulation of information in working memory". In: *Nature Neuroscience* 22.7, pp. 1159–1167. DOI: 10.1038/s41593-019-0414-3. URL: https://doi.org/10.1038/s41593-019-0414-3.

Masse, Nicolas Y. et al. (June 2019b). "Circuit mechanisms for the maintenance and manipulation of information in working memory". In: *Nature Neuroscience* 22.7, pp. 1159–1167. DOI: 10.1038/s41593-019-0414-3.

Mikolov, T. et al. (2013). "Distributed representations of words and phrases and their compositionality". In: *Proc. of NIPS*, pp. 3111–3119.

Miller, Earl K. and Jonathan D. Cohen (Mar. 2001). "An Integrative Theory of Prefrontal Cortex Function". In: *Annual Review of Neuroscience* 24.1, pp. 167–202. DOI: 10.1146/annurev.neuro.24.1.167.

Miller, George A. (1956). "The magical number seven, plus or minus two: some limits on our capacity for processing information." In: *Psychological Review* 63.2, pp. 81–97. DOI: 10.1037/h0043158. URL: https://doi.org/10.1037/h0043158.

Miller, George A., Eugene Galanter, and Karl H. Pribram (1960). *Plans and the structure of behavior*. Henry Holt and Co. DOI: 10.1037/10039-000. URL: https://doi.org/10.1037/10039-000.

Monchi, Oury et al. (Oct. 2001). "Wisconsin Card Sorting Revisited: Distinct Neural Circuits Participating in Different Stages of the Task Identified by Event-Related Functional Magnetic Resonance Imaging". In: *The Journal of Neuroscience* 21.19, pp. 7733–7741. DOI: 10.1523/jneurosci.21-19-07733.2001. URL: https://doi.org/10.1523/jneurosci.21-19-07733.2001.

Mongillo, G., O. Barak, and M. Tsodyks (Mar. 2008a). "Synaptic Theory of Working Memory". In: *Science* 319.5869, pp. 1543–1546. DOI: 10.1126/science.1150769. URL: https://doi.org/10.1126/science.1150769.

– (Mar. 2008b). "Synaptic Theory of Working Memory". In: *Science* 319.5869, pp. 1543–1546. DOI: 10.1126/science.1150769.

Mossakowski, T., R. Diaconescu, and M. Glauer (2019). "Towards logics for neural conceptors". In: *J of Applied Logics* 6.4, pp. 725–744.

Murdock, Bennet B. (1968). "Serial order effects in short-term memory." In: *Journal of Experimental Psychology* 76.4, Pt.2, pp. 1–15. DOI: 10.1037/h0025694. URL: https://doi.org/10.1037/h0025694.

Nachstedt, Timo and Christian Tetzlaff (May 2017). "Working Memory Requires a Combination of Transient and Attractor-Dominated Dynamics to Process Unreliably Timed Inputs". In: *Scientific Reports* 7.1. DOI: 10.1038/s41598-017-02471-z.

Neelakantan, Arvind, Quoc V. Le, and Ilya Sutskever (2015). *Neural Programmer: Inducing Latent Programs with Gradient Descent*. arXiv: 1511.04834 [cs.LG].

Oberauer, Klaus (2009). "Chapter 2 Design for a Working Memory". In: *The Psychology of Learning and Motivation*. Elsevier, pp. 45–100. DOI: 10.1016/s0079-7421(09)51002-x. URL: https://doi.org/10.1016/s0079-7421(09)51002-x.

Oberauer, Klaus and Reinhold Kliegl (Mar. 2001). "Beyond resources: Formal models of complexity effects and age differences in working memory". In: *European Journal of Cognitive Psychology* 13.1-2, pp. 187–215. DOI: 10.1080/09541440042000278. URL: https://doi.org/10.1080/09541440042000278.

Oberauer, Klaus et al. (Sept. 2018). "Benchmarks for models of short-term and working memory." In: *Psychological Bulletin* 144.9, pp. 885–958. DOI: 10.1037/bul0000153. URL: https://doi.org/10.1037/bul0000153.

O'Reilly, Randall C. and Michael J. Frank (Feb. 2006a). "Making Working Memory Work: A Computational Model of Learning in the Prefrontal Cortex and Basal Ganglia". In: *Neural Computation* 18.2, pp. 283–328. DOI: 10.1162/089976606775093909. URL: https://doi.org/10.1162/089976606775093909.

– (Feb. 2006b). "Making Working Memory Work: A Computational Model of Learning in the Prefrontal Cortex and Basal Ganglia". In: *Neural Computation* 18.2, pp. 283–328. DOI: 10.1162/089976606775093909.

Pascanu, Razvan and Herbert Jaeger (Mar. 2011a). "A neurodynamical model for working memory". In: *Neural Networks* 24.2, pp. 199–207. DOI: 10.1016/j.neunet.2010.10.003. URL: https://doi.org/10.1016/j.neunet.2010.10.003.

– (Mar. 2011b). "A neurodynamical model for working memory". In: *Neural Networks* 24.2, pp. 199–207. DOI: 10.1016/j.neunet.2010.10.003.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2012). *On the difficulty of training Recurrent Neural Networks*. arXiv: 1211.5063 [cs.LG].

Peterson, Lloyd and Margaret Jean Peterson (1959). "Short-term retention of individual verbal items." In: *Journal of Experimental Psychology* 58.3, pp. 193–198. DOI: 10.1037/h0049234. URL: https://doi.org/10.1037/h0049234.

Pham, D.T and D Karaboga (Apr. 1999). "Training Elman and Jordan networks for system identification using genetic algorithms". In: *Artificial Intelligence in Engineering* 13.2, pp. 107–117. DOI: 10.1016/s0954-1810(98)00013-2. URL: https://doi.org/10.1016/s0954-1810(98)00013-2.

Pribram, KH et al. (1964). "A progress report on the neurological processes disturbed by frontal lesions in primates". In: *The frontal granular cortex and behavior*. McGraw-Hill New York, pp. 28–55.

Rigotti, Mattia et al. (May 2013a). "The importance of mixed selectivity in complex cognitive tasks". In: *Nature* 497.7451, pp. 585–590. DOI: 10.1038/nature12160. URL: https://doi.org/10.1038/nature12160.

– (May 2013b). "The importance of mixed selectivity in complex cognitive tasks". In: *Nature* 497.7451, pp. 585–590. DOI: 10.1038/nature12160.

Rodan, A and P Tino (Jan. 2011). "Minimum Complexity Echo State Network". In: *IEEE Transactions on Neural Networks* 22.1, pp. 131–144. DOI: 10.1109/tnn.2010.2089641. URL: https://doi.org/10.1109/tnn.2010.2089641.

Rohrer, Doug and John T. Wixted (Sept. 1994). "An analysis of latency and interresponse time in free recall". In: *Memory & Cognition* 22.5, pp. 511–524. DOI: 10.3758/bf03198390. URL: https://doi.org/10.3758/bf03198390.

Romo, Ranulfo and Emilio Salinas (Mar. 2003). "Flutter Discrimination: neural codes, perception, memory and decision making". In: *Nature Reviews Neuroscience* 4.3, pp. 203–218. DOI: 10.1038/nrn1058. URL: https://doi.org/10.1038/nrn1058.

Romo, Ranulfo et al. (June 1999). In: *Nature* 399.6735, pp. 470–473. DOI: 10.1038/20939.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1985). *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science.

Schaetti, Nils, Michel Salomon, and Raphael Couturier (Aug. 2016). "Echo State Networks-Based Reservoir Computing for MNIST Handwritten Digits Recognition". In: *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*. IEEE. DOI: 10.1109/cse-euc-dcabes.2016.229.

Schäfer, Anton Maximilian and Hans-Georg Zimmerman (Aug. 2007). "RECURRENT NEURAL NETWORKS ARE UNIVERSAL APPROXIMATORS". In: *International Journal of Neural Systems* 17.04, pp. 253–263. DOI: 10.1142/s0129065707001111. URL: https://doi.org/10.1142/s0129065707001111.

Schuessler, Friedrich et al. (Feb. 2020a). "Dynamics of random recurrent networks with correlated low-rank structure". In: *Physical Review Research* 2.1. DOI: 10.1103/physrevresearch.2.013111. URL: https://doi.org/10.1103/physrevresearch.2.013111.

Schuessler, Friedrich et al. (2020b). "Dynamics of random recurrent networks with correlated low-rank structure". In: *Physical Review Research* 2.1, p. 013111.

Semon, Richard Wolfgang (1921). *The mneme*. G. Allen & Unwin Limited.

Seung, H. S. (Nov. 1996). "How the brain keeps the eyes still". In: *Proceedings of the National Academy of Sciences* 93.23, pp. 13339–13344. DOI: 10.1073/pnas.93.23.13339.

Seung, H. Sebastian (1998). "Learning Continuous Attractors in Recurrent Networks". In: *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems 10*. NIPS '97. Denver, Colorado, USA: MIT Press, pp. 654–660. ISBN: 0-262-10076-2. URL: http://dl.acm.org/citation.cfm?id=302528.302766.

Shallice, T. and Elizabeth K. Warrington (May 1970). "Independent Functioning of Verbal Memory Stores: A Neuropsychological Study". In: *Quarterly Journal of Experimental Psychology* 22.2, pp. 261–273. DOI: 10.1080/00335557043000203. URL: https://doi.org/10.1080/00335557043000203.

Siegler, Robert S. and Julie L. Booth (Mar. 2004). "Development of Numerical Estimation in Young Children". In: *Child Development* 75.2, pp. 428–444. DOI: 10.1111/j.1467-8624.2004.00684.x. URL: https://doi.org/10.1111/j.1467-8624.2004.00684.x.

Soto, David, Teemu Mäntylä, and Juha Silvanto (Nov. 2011). "Working memory without consciousness". In: *Current Biology* 21.22, R912–R913. DOI: 10.1016/j.cub.2011.09.049. URL: https://doi.org/10.1016/j.cub.2011.09.049.

Stokes, Mark G. (July 2015a). "'Activity-silent' working memory in prefrontal cortex: a dynamic coding framework". In: *Trends in Cognitive Sciences* 19.7, pp. 394–405. DOI: 10.1016/j.tics.2015.05.004. URL: https://doi.org/10.1016/j.tics.2015.05.004.

– (July 2015b). "'Activity-silent' working memory in prefrontal cortex: a dynamic coding framework". In: *Trends in Cognitive Sciences* 19.7, pp. 394–405. DOI: 10.1016/j.tics.2015.05.004.

Strock, Anthony, Xavier Hinaut, and Nicolas P. Rougier (Jan. 2020). "A Robust Model of Gated Working Memory". In: *Neural Computation* 32.1, pp. 153–181. DOI: 10.1162/neco_a_01249. URL: https://doi.org/10.1162/neco_a_01249.

Strock, Anthony, Nicolas Rougier, and Xavier Hinaut (2020). *Transfer between long-term and short-term memory using Conceptors*. arXiv: 2003.11640 [cs.NE].

Strock, Anthony, Nicolas P. Rougier, and Xavier Hinaut (July 2018). "A Simple Reservoir Model of Working Memory with Real Values". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. Sao Paulo, Brazil: IEEE. DOI: 10.1109/ijcnn.2018.8489262.

Sukhbaatar, Sainbayar et al. (2015). "End-To-End Memory Networks". In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al. Curran Associates, Inc., pp. 2440–2448. URL: http://papers.nips.cc/paper/5846-end-to-end-memory-networks.pdf.

Sussillo, D (2014). "Neural circuits as computational dynamical systems". In: *Current opinion in neurobiology* 25, pp. 156–163.

Sussillo, David and L.F. Abbott (Aug. 2009). "Generating Coherent Patterns of Activity from Chaotic Neural Networks". In: *Neuron* 63.4, pp. 544–557. DOI: 10.1016/j.neuron.2009.07.018.

Sussillo, David and Omri Barak (Mar. 2013a). "Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks". In: *Neural Computation* 25.3, pp. 626–649. DOI: 10.1162/neco_a_00409. URL: https://doi.org/10.1162/neco_a_00409.

– (Mar. 2013b). "Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks". In: *Neural Computation* 25.3, pp. 626–649. DOI: 10.1162/neco_a_00409.

Tio, Peter (2020). "Dynamical systems as temporal feature spaces". In: *Journal of Machine Learning Research* 21.44, pp. 1–42.

Tulving, Endel et al. (1972). "Episodic and semantic memory." In:

Unsworth, Nash and Randall W. Engle (Jan. 2006). "Simple and complex memory spans and their relation to fluid abilities: Evidence from list-length effects". In: *Journal of Memory and Language* 54.1, pp. 68–80. DOI: 10.1016/j.jml.2005.06.003. URL: https://doi.org/10.1016/j.jml.2005.06.003.

Uytun, Merve Cikili (Oct. 2018). "Development Period of Prefrontal Cortex". In: *Prefrontal Cortex*. InTech. DOI: 10.5772/intechopen.78697. URL: https://doi.org/10.5772/intechopen.78697.

Velichkovsky, Boris B. (Oct. 2017). "Consciousness and working memory: Current trends and research perspectives". In: *Consciousness and Cognition* 55, pp. 35–45. DOI: 10.1016/j.concog.2017.07.005. URL: https://doi.org/10.1016/j.concog.2017.07.005.

Verstraeten, D. et al. (Apr. 2007). "An experimental unification of reservoir computing methods". In: *Neural Networks* 20.3, pp. 391–403. DOI: 10.1016/j.neunet.2007.04.003.

Voelker, Aaron, Ivana Kajić, and Chris Eliasmith (2019). "Legendre Memory Units: Continuous-Time Representation in Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., pp. 15570–15579. URL: http://papers.nips.cc/paper/9689-legendre-memory-units-continuous-time-representation-in-recurrent-neural-networks.pdf.

Walt, Stéfan van der, S Chris Colbert, and Gaël Varoquaux (Mar. 2011). "The NumPy Array: A Structure for Efficient Numerical Computation". In: *Computing in Science & Engineering* 13.2, pp. 22–30. DOI: 10.1109/mcse.2011.37.

Watanabe, K. and S. Funahashi (Aug. 2007). "Prefrontal Delay-Period Activity Reflects the Decision Process of a Saccade Direction during a Free-Choice ODR Task". In: *Cerebral Cortex* 17.suppl 1, pp. i88–i100. DOI: 10.1093/cercor/bhm102. URL: https://doi.org/10.1093/cercor/bhm102.

Watanabe, Kei and Shintaro Funahashi (Mar. 2014). "Neural mechanisms of dual-task interference and cognitive capacity limitation in the prefrontal cortex". In: *Nature Neuroscience* 17.4, pp. 601–611. DOI: 10.1038/nn.3667. URL: https://doi.org/10.1038/nn.3667.

Wei, Z., X.-J. Wang, and D.-H. Wang (Aug. 2012). "From Distributed Resources to Limited Slots in Multiple-Item Working Memory: A Spiking Network Model with Normalization". In: *Journal of Neuroscience* 32.33, pp. 11228–11240. DOI: 10.1523/jneurosci.0735-12.2012.

Weston, Jason, Sumit Chopra, and Antoine Bordes (2014). *Memory Networks*. arXiv: 1410.3916 [cs.AI].

Wikipedia contributors (2020). *Wisconsin Card Sorting Test — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Wisconsin_Card_Sorting_Test&oldid=962466163. [Online; accessed 7-September-2020].

Wilken, Patrick and Wei Ji Ma (Dec. 2004). "A detection theory account of change detection". In: *Journal of Vision* 4.12, p. 11. DOI: 10.1167/4.12.11. URL: https://doi.org/10.1167/4.12.11.

Yang, Guangyu Robert et al. (Jan. 2019). "Task representations in neural networks trained to perform many cognitive tasks". In: *Nature Neuroscience* 22.2, pp. 297–

306. DOI: 10.1038/s41593-018-0310-2. URL: https://doi.org/10.1038/s41593-018-0310-2.

Yu, Yong et al. (July 2019). "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures". In: *Neural Computation* 31.7, pp. 1235–1270. DOI: 10.1162/neco_a_01199. URL: https://doi.org/10.1162/neco_a_01199.

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

Zhou, Chunting et al. (2015). *A C-LSTM Neural Network for Text Classification*. arXiv: 1511.08630 [cs.CL].

Zipser, D et al. (Aug. 1993). "A spiking network model of short-term active memory". In: *The Journal of Neuroscience* 13.8, pp. 3406–3420. DOI: 10.1523/jneurosci.13-08-03406.