Modeling serial recall in working memory with recurrent neural networks

Auteur : Stordeur, Lucas
Promoteur(s) : Sacré, Pierre
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil biomédical, à finalité spécialisée
Année académique : 2024-2025
URI/URL : http://hdl.handle.net/2268.2/23282

UNIVERSITE DE LIEGE
FACULTE DES SCIENCES APPLIQUEES

Modeling serial recall in working memory with recurrent neural networks

Stordeur Lucas

Thesis presented to obtain the degree of :
Master of Science in Biomedical Engineering

Thesis supervisors :
Pr. Sacré Pierre, Pr. Majerus Steve

Academic year: **2024 - 2025**

Acknowledgments

I would like to express my deepest thanks to my supervisors, Pr. Pierre Sacré and Pr. Steve Majerus, who guided me through all of this journey. Both of your perspectives created very interesting discussions at the edge of psychology and computational modeling, which made my interest in this field grow. Your interest, availability and support have been a driving force throughout this year. I am looking forward to our future work together.

I would also like to thank Professor Matthew Botvinick, who took the time to answer our questions and helped us understand the direction our work was taking. This exchange was a tremendous help in moving this work forward. Thanks to Antoine Debor and Florent De Geeter for the discussions we have about the RNNs.

Thanks to my parents, Philippe and Laurence, and sisters, Axelle and Alice, for their constant support throughout my academic career and daily life, and for brightening up my days when I come home.

Thanks to my friends, Clémence, Ahmed, Julien, Fanny and Linpha, who made this five years much funnier. But also to Quentin and Aurélie for all the time we spent together.

Summary

In psychology, various tasks are designed to study cognitive functions such as language processing, long-term memory, and working memory. Immediate serial recall is one such task, where participants are presented with a sequence of items (e.g., words or letters) and must recall them in the same order after the sequence is presented. This task provides valuable insights into working memory mechanisms.

Computational modeling of psychological experiments helps improve our understanding of working memory by replicating experimental data. Different models exist for serial recall, each based on distinct mechanisms and assumptions that explain various observed behaviors. The quality of these models is usually assessed by comparing their predictions with real experimental data. Among these models, RNNs stand out due to their dynamical properties and strong empirical support from neuroscience. An RNN consists of three main layers: an input layer, where sequence items are presented; a hidden layer, where information is processed; and an output layer, which generates predictions. A key feature of RNNs is their recurrent connections, allowing past inputs to influence future predictions, making them well-suited for capturing temporal relationships in sequences.

The first objective of my thesis is to construct an RNN capable of reproducing three key effects observed in serial recall experiments: length-dependent performance (recall accuracy decreases sigmoidally as sequence length increases), primacy and recency effects (better recall of the first and last items in a sequence), transposition errors (recall mistakes typically involve swapping items) and to explore the reproduction of many other effects characteristic of this task. To achieve this, I formally modeled the serial recall task and then trained the RNN. Unlike standard deep learning approaches that optimize for maximum accuracy, our model is trained to match the recall performance observed in human participants. This requires two key modifications: (1) training the network only until it reaches human-like recall accuracy and (2) varying the dataset size to prevent overfitting to specific sequences. We will show that despite the difficulties that many encountered to construct such a model, recurrent neural networks are able to reproduce different empirical benchmarks.

The second objective is to explore the internal dynamics of the networks. Specifically, we investigate whether modifying the cell dynamics to be more biologically plausible improves the model alignment with experimental data. The hidden layer activation dynamics—determined by the interaction between current inputs and past hidden states—are progressively refined. I begin with a simple Elman network, then introduce more complex architectures: Gated Recurrent Unit (GRU) cells, which incorporate a forgetting mechanism, Bistable Recurrent Cells (BRCs), which introduce bistability properties. We will show that more complex and biologically plausible allows to achieve better results on the empirical benchmarks.

Finally, as this model has not yet been widely explored, we explored its behavior by investigating how the variability can be included in the curves, as they are a central characteristic of the human data, but is not well explored in model. In addition, the modelization of the inter-patient variability. In addition, saturation of model training will be investigated and will allow to create a more plausible model. All of these contributes to the better understanding of the model.

In conclusion, through these refinements, we aimed to bridge the gap between artificial and biological neural networks, contributing to a deeper understanding of working memory.

Contents

Introduction

Context

In psychology, many experiments are used to try to better understand the functioning of memory. Of all these tasks, serial recall is one of the most widely used to study working memory and consists of recalling sequences of items.

However, even with this task, this is still not easy to understand the functioning of working memory. For this different models of the tasks have been developed.

Among all of these, recurrent neural networks are a type of model not well studied as many struggled to reproduce the empirical benchmarks with such a model. Therefore, being able to develop such a model would allow to better understand it and assess its validity to model the serial recall task.

Objective

This thesis is composed of three main objectives to try to better understand the task of serial recall using a recurrent neural network model.

- The first objective of this work is to recreate a recurrent neural network model that is able to learn the serial recall task correctly. For this, the task will be modeled such that it can be used to train the network. Then, the architecture of the network will be developed such that the network is able to recall correctly the sequences presented as inputs to the network, once its training is finished. To assess the validity of the model, the predictions of the model will be compared to different empirical benchmarks. This is a really important point as many researchers tried to implement a recurrent neural network model of serial recall, but with no success as they were not able to reproduce the typical experimental effects observed in serial recall.
- Once the first objective has been reached, the dynamic of the recurrent cells will be investigated. In fact, the only literature studying this type of model used a simple Elman cell, which is the most basic dynamic available for the recurrent network. Throughout the last years, other cells have been developed to solve computational observed in this kind of network like the GRU cells, while others have been to reduce the gap between real and artificial neurons, by introducing cells that are biologically inspired, the BRC and nRBC cells. The different cells will be used to replace the Elman cell and their impact on the model predictions will be characterized. It will then be used to assess whether cells that are more neurally possible would lead to predictions closer to human data.
- The last objective of this master thesis is to investigate the global behavior of the model to improve it to make it more plausible and to assess its limitations. For this, we will first investigate the variability of the serial position recall curves, which is a central aspect in the empirical benchmarks but rarely reproduced in the different models. The saturation of the model will be discussed, as it plays a central role in making the model more plausible. Finally, the generalization of the model to

new items has also been considered. All these points are essential as they allow to better understand the model.

Document Organization

This master thesis is composed of four main parts.

The first part will give a short introduction on memory before focusing on the serial recall task which will interest us for the rest of the work. For this the task and its effects will be described and the different models that have been developed to analyze it.

In the second part, we focus on the background necessary to understand correctly the Recurrent neural network model developed.

Third part focus on the implementation of the model, the reproduction of the main experimental results and the comparison of the different cells.

Finally, the last part will discuss the model limitations and perspectives.

Part I

Working memory: the case of serial recall task

Chapter 1

Memory

Memory can be defined as the capacity to store and retrieve information (*Zlotnik and Vansintjan, 2019*). In addition, it has the ability to adapt its behavior to previous experiences. Memory is therefore essential for human beings, as it is needed all the time in everyday life. Indeed, whether it is remembering the words to a popular song after hearing its first notes, recalling people's names, remembering how to ride a bike or following directions, all these applications require the involvement of memory (*Wingfield and Byrnes, 1981*).

The storage of all this information is done through different processes. In most organisms, synaptic consolidation is used to store information in the long term (*Bramham and Messaoudi, 2005*), but system consolidation is also an important process in complex organisms as it processes, moves and stores information more permanently (*Frankland and Bontempi, 2005*).

1.1 Types of memory

Describing memory is difficult as it can take many different forms. To distinguish these different memories, different criteria have been created to classify them.

Short and long-term memory

Memory has different timescales; therefore, a first natural classification is to assess whether memory is short- or long-term. Short-term memory retains information for a brief period of time, typically from seconds to minutes. This memory, also known as working memory, contains all the information retrieved from other types of memory as it is used. On the other hand, long-term memory has the ability to store information over periods of days to years (*Cowan, 2008*).

Declarative and non-declarative memory

Long-term memory can be further categorized by assessing whether the memory is declarative or not. Declarative memory, also known as explicit memory, refers to knowledge or experiences that can be consciously remembered, while this is made unconsciously in non-declarative memory (*Squire, 1992*).

Types of declarative memory

The declarative memory can be decomposed into two distinct memory systems: the semantic memory and the episodic memory. Semantic memory is essential for the acquisition, representation, and processing of conceptual information that could be shared by a significant number of individuals (Saumier and Chertkow, 2002). The episodic memory encodes the daily personal experiences and is unique to each individual (*Tulving, 2002*).

Types of non-declarative memory

The non-declarative memory is composed of four different systems: priming, conditioning, procedural, and non-associative learning. Priming is a phenomenon where the exposure to specific stimuli has an unconscious influence on a following related stimulus (*Ratcliff and McKoon, 1988*). Conditioning consists of the association of a powerful stimulus with a neutral stimulus such that the presence of the neutral stimulus will activate the other stimulus (*Bouton and Moody, 2004*). Procedural memory refers to the acquisition of cognitive and sensorimotor habits and skills that have been learned through repetition to consolidate memory (*Lee, 2004*). Finally, non-associative learning refers to the change in behavior over time towards a stimulus, in the absence of any association with other stimuli that would induce such a change. This last type of memory is based on frequency. Habituation and sensitization are two important phenomena for this type of memory (*Blumstein, 2016*).

1.2 Working Memory

Working memory is the ability to store and retrieve information after a short period of time, typically on the order of a few seconds. This type of memory can only store a small amount of information to process them, but is able to store information while performing other complex tasks. From a neuroscience perspective, it has been shown that the fronto-parietal regions of the brain were activated by working memory (*Chai et al., 2018*).

Typical tasks involving working memory are recalling a phone number after hearing him, following a recipe, doing mental calculus, etc. During these tasks, a human can remember only a limited number of items. The typical number of elements is equal to seven plus or minus two (*Baddeley and Hitch, 1974*).

1.2.1 Neuronal mechanisms

It has been suggested by Baddeley & Hitch that the working memory is composed of one central executive system and two loops (*Miller, 1994*). The central executive is the main unit that is responsible for the regulation of the integration of information coming from different sources, the allocation of available resources, and the coordination of different cognitive processes.

The two loops, the phonological loop and the visuo-spatial sketch pad, treat different types of information. The phonological loop stores verbal information while the visuo-spatial sketchpad maintains and manipulates the spatial information. These three elements allow to manipulate the information coming in the working memory. This representation of working memory was later extended by adding an episodic buffer, which maintains the representation of the information (*Baddeley, 2000*) .

Chapter 2

The serial recall task

2.1 Introduction

Serial recall is one of the most widely used task in psychology and neuropsychology to study short-term memory (Kreutzer et al., 2011). In this task, a sequence of items, e.g. digits, letters, words, etc., is presented to the participants. These items are usually presented one at a time, for a short duration, either visually or acoustically. After a short period of time, the participants have to recall the different items of the sequence in a specific order. This task is different from free recall, where the participants can recall the items of the sequences in the order they come to mind.

The most common version of the task is immediate forward serial recall, in which the participants have to recall the items in the order they were presented immediately after the presentation of the sequence (*Cowan et al., 2004*). However, many variations of the task are also used to study specific observed effects and aspects of the task. These variations can come from the type of items presented to the participants, the presentation modalities, or the type of serial recall (*Greene, 1989*).

The task can also be made more complex by introducing distractors between the learning and recall phases. The distractors consists of a different task that the participant has to do such that its brain is not focused on the the items of the sequence (*Colom et al., 2006*). However, as this situation is much more complex, this will not be considered in this work.

2.2 Serial and free recall

Another very used task in psychology, similar to serial recall, is free recall. The only difference between these two is that, in free recall, the participants can recall the items of the sequence without any constraint on the recall order (*Hertzog et al., 2010*). The main performance measurement for this task is the number of items recalled correctly, which is used to study the impact of different parameters on the performance, e.g. list length, type of items, etc. In this task, participants generally initiate the recall of the sequence with the terminal items of the list, leading to a recency effect. Different studies have highlighted that the differences between these two paradigms are mainly due to their differences in recall instructions (*Waugh, 1961*; *Klein et al., 2005*).

2.3 Experimental effects

In a serial recall, many different effects can be observed which are characteristic of the task. These effects are essentials as they allow to better understand the functioning of working memory. This section aims to introduce the major observed effects that will be studied in this work.

2.3.1 Working memory performance as function of the sequence length

The length of the sequence of items presented to the participants has a strong impact on performance during recall. It is generally accepted that people are able to recall lists of up to seven items after a single presentation of the sequence (*Baddeley and Hitch, 1974*).

This characteristic is referred to the memory span, which is the maximum number of items that a person can recall immediately in the correct order, on more than 50% of the different trials. This effect is of crucial importance to assess working memory capability of individuals (*Blankenship, 1938*). Many factors affect memory span, some of which are extrinsic, e.g. rate of presentation, fatigue, distraction, practice, effect of drugs, etc., and others are intrinsic, e.g. sex, age, permanent pathological condition, etc. Controlling extrinsic factors is of high importance to obtain high reliability between different tests. The influence of the factors on the performance is usually assessed using different types of items and presentation modalities (*Dempster, 1981*).

Once the memory span is reached, longer sequences exhibit a strong decrease in accuracy to rapidly reach performances close to 0%. A typical curve of this effect is represented in Figure 2.1. This curve has a sigmoid shape, where for short sequences, the accuracy is close to the saturation of 100% performance, as the sequence is not long enough to reach the limit of memory capability. Accuracy only starts to decrease for sequence length close to the memory span value. Once this limit crossed, performance rapidly decreases towards zero as it becomes really difficult to recall entirely long sequences.

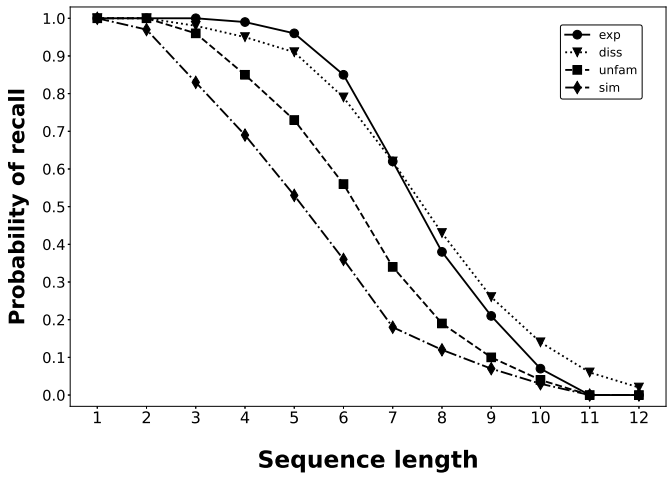


Figure 2.1: Evolution of the working memory performance with the length of the sequence for different types of items. The x-axis indicates the length of the sequence, while the y-axis indicates the probability that the a sequence of this length will be correctly recalled, figure adapted from *Guilford and Dallenbach, 1925*.

This first effect is therefore of the utmost importance to show the limit of working memory in the recall task. The figure also showed the influence of various type of items on the curves, where similar items show the most rapid decline.

2.3.2 Serial position recall curve

The second effect emerges from the performance of a fixed length sequence. When a sequence of items is presented to the participants, the recall performance does not tend to be uniform for all the items. Instead, some items will show better recall performance than others. However, the variations of performance do not occur randomly, but according to a bow-shaped curve, represented in figure 2.2.

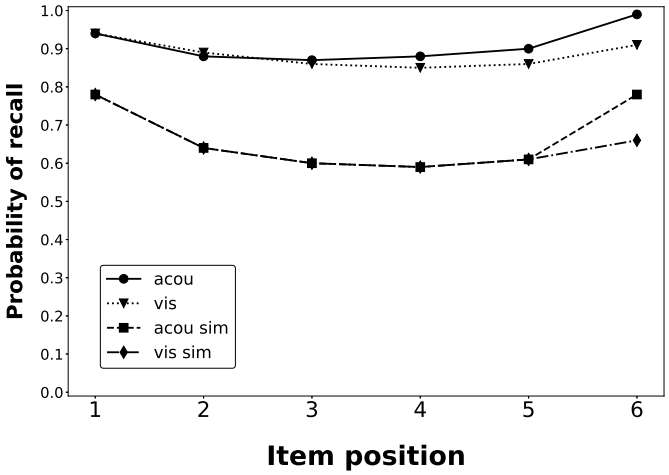


Figure 2.2: Serial position recall curves for different types of items. The x-axis indicates the position of the item, while the y-axis indicates the probability that the item is correctly recalled, figure adapted from *Burgess and Hitch, 1999*.

This recall curve presents better performance for the first and last items of the sequence. This is the consequence of two different effects, the primacy effect and the recency effect.

Primacy effect is a cognitive bias that results in a better recall performance for items at the beginning of the list. Indeed, first items are more likely to be stored in the long-term memory, and so less processing power is needed for the brain to rehearse and recall the first item of the sequence than the following ones. In addition to being present in the serial recall task, this effect is well known in psychology, as it can affect decision-making because the first information presented is more easily remembered than later information (*Ahnke, 1965*).

Recency effect is the tendency to better remember the last items presented in the sequence. An explanation to this phenomenon is that the last items are still in working memory when recall occurs. This effect increases when too much information is presented too quickly and is reduced when memorization is coupled with other tasks or when delay between list presentation and rehearsal becomes sufficiently large (*Ahnke, 1965*).

These two effects can be affected by pathological conditions. In Alzheimer's disease, the memory alteration results in a reduced primacy effect and the absence of recency effect, whereas people suffering from amnesia do not present a primacy effect, as they have poor abilities to form permanent long-term memories (*Ahnke, 1965*).

The recall performance for items in the middle of the list is the worst, as they benefit neither from primacy nor recency effects. For sequences of six items, the recall curve is quite symmetrical, as represented in Figure 2.2, showing recency and primacy effect of the same size. This is because the sequence is sufficiently short such that recall performance does not decrease a lot. For longer sequences, the recall curve is no longer symmetric. The primacy effect still allows to get good recall performances for the first items. Then, performances decrease with the position of the item until the last items, where the recency effects allow a small increase of the performance (*Burgess and Hitch, 1999*).

This curve and more particularly the two effects discussed are particularly important to understand as they are used on a daily basis, more particularly in marketing applications, social interactions, etc.

2.3.3 Type of errors and effect of age

When recalling a sequence, all the items are not correctly recalled, resulting in the two previous curves. However, it is also important to analyze what item is recalled when the participant does not give the correct item. For this, different types of errors can be introduced.

The first type of error is omission errors and consists of the absence of response for a particular serial position in the sequence. This type of error appears when there is a lot of uncertainty between different items. Intrusion errors, the second type of error, consist of recalling an item that was not in the original sequence presented. The third type of error is repetition errors, which are recalling multiple times the same item of the sequence. The repetitions can either be consecutive or separated in the recall sequence. Finally, the last type of errors is transposition and consists of recalling a list item in an incorrect serial position. This could correspond to inversion of two items in the sequence or to larger movements in the recall sequence, where one item is moved and affects the recall position of multiple items of the sequence (*Guilford and Dallenbach, 1925*).

All these different errors do not appear with the same frequency during the recall of the sequences. Indeed, some error types are produced more often than others. The proportion of errors for an adult is represented in figure 2.3.

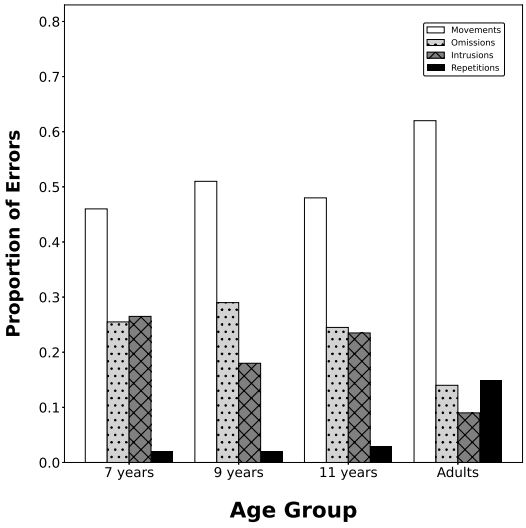


Figure 2.3: Proportion of the types of error for different participants. The x-axis indicates the age of the participants, while the y-axis indicates the corresponding proportion of errors, Figure adapted from *McCormack et al., 2000*.

In addition to giving the proportion of error types for adult participants, Figure 2.3 presents the same analyses for participants of age 7, 9 and 11 and highlighted the impact of age on the effects obtained (*Kay, 2001*). First, younger participants tend to achieve a lower global accuracy compared to adults, as memory undergoes developmental changes over time. Then, transposition errors become the most predominant error type for all the participants' ages. However, the proportion of other errors changes with age, with fewer repetition errors for younger participants.

2.3.4 Transposition gradients

Last section introduced the different types of errors present in serial recall experiments, but deeper analyses can be conducted on the most predominant type of errors, transposition errors. As transposition errors

correspond to movements of items in the recall sequence, an important and more in-depth analysis is to assess the distance between the moved item and its original position. Generally, humans are more likely to change the item's position to its neighbors rather than to positions that are far away. This results in a transposition gradient, with higher frequencies for small distances, as depicted in figure 2.4.

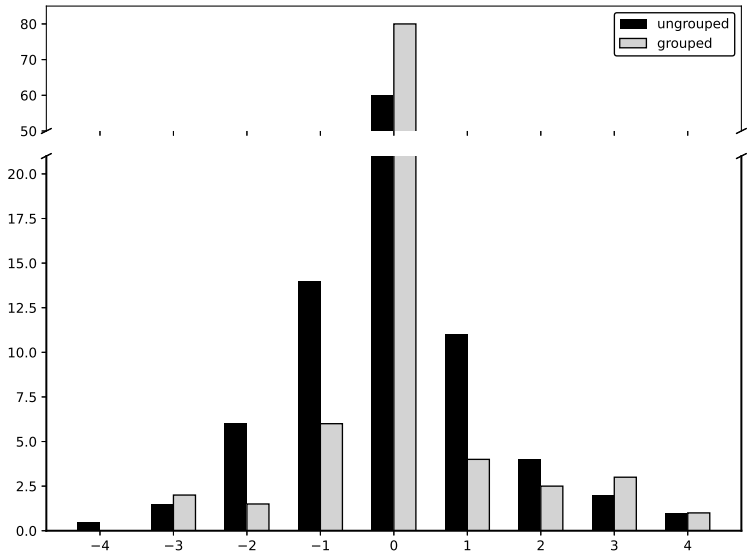


Figure 2.4: Effects of temporal grouping on the transposition gradients, for items grouped by groups of three or ungrouped, Figure adapted from *Burgess and Hitch, 1999*.

The transposition distances are also a developmental effect. The steepness of the gradient tends to increase with age, meaning that adults tend to be more likely to recall items close to their true position, as represented in figure 2.5. In addition, temporal grouping of items also affect the experimental results obtained (*Hitch, 2010*).

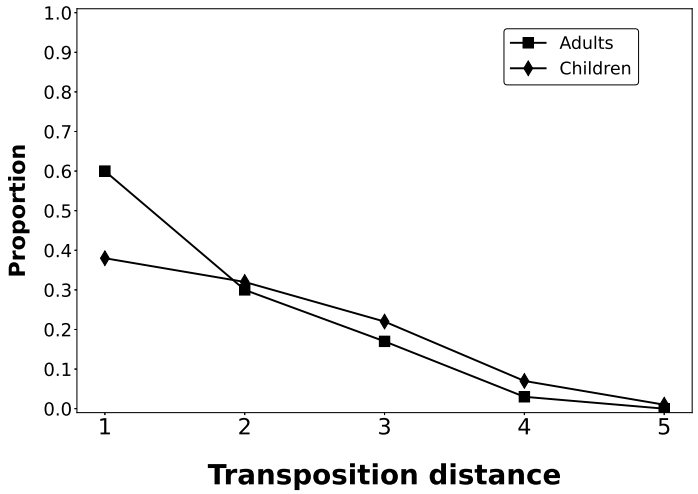


Figure 2.5: Transposition gradients. The x-axis indicates the position in the target list, the clusters represent the position of this item in the predicted list, figure adapted from *Henson, 1998*.

A last way to analyze transposition errors is to analyze, for a sequence of fixed length, for each true item position, the position of the same item in the recall sequence. This allows to analyze to different behavior, the recall performance as function of the item position, by looking the proportion when the recall position is identical to the true item position, and the evolution of the transposition errors as a function

of item positions (*Henson, 1996*). This type of results are represented in figure 2.6.

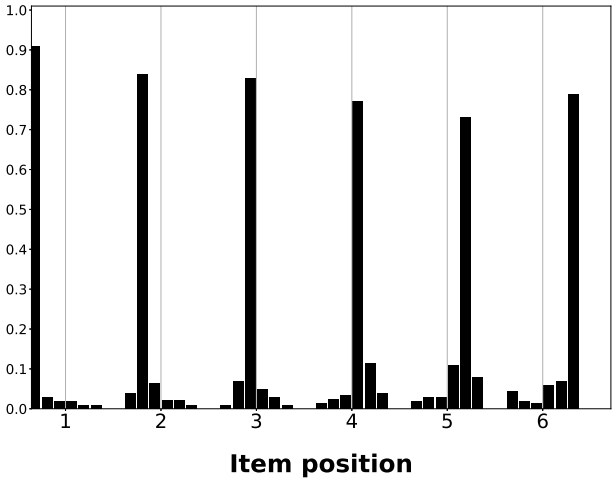


Figure 2.6: Evolution of the working memory performance with the length of the sequence. The x-axis indicates the length of the sequence, while the y-axis indicates the probability that the a sequence of this length will be correctly recalled, figure adapted from *Henson, 1996*.

This type of curves resumes the effects previously explained, with the recall performance presenting recency and primacy effects, leading to a bow-shaped performance curve and transposition errors that are more frequent for small distances than for larger ones.

2.3.5 Confusability

One important aspect of serial recall is its sensibility to similarity between items in the sequence. To assess this impact, acoustic similarity is used to create four types of sequences, sequences composed of purely confusable items, purely non-confusable items, alternation of confusable and non-confusable, starting with a confusable item (AC) and starting with a non-confusable item (AN) (*Baddeley, 1968*). The type of sequences presented have a considerable impact on the recall curve obtained, as represented in Figure 2.7.

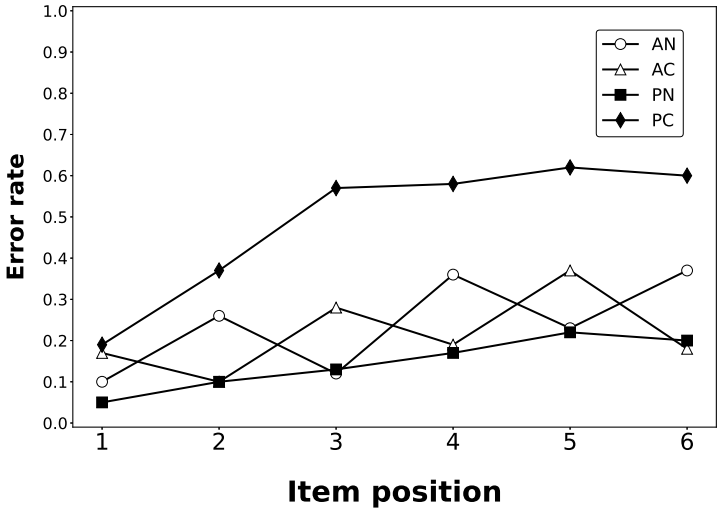


Figure 2.7: Serial position recall curves of lists composed of pure confusable items (PC), Pure non-confusable items (PN), alternation confusable non-confusable (AC) and alternation non-confusable confusable (AN), figure adapted from *Baddeley, 1968*.

As represented by the experimental data, the worst recall performances are obtained for list off pure confusable items, as it is more easy to confuse between two items, whereas non-confusable items lists have the best performance. Between the two are the alternation of confusable and non-confusable elements in the sequences. These sequences present a sawtooth pattern, where the confusable items are less easily recalled than the non-confusable ones. Despite the confusability effect, the two typical effects of the recall curve, i.e. recency and primacy, can still be observed.

2.3.6 Bigram frequency

A bigram is a pair of consecutive items. These items can be words, syllables, letters, etc. This concept is essential to better understand how regularities in sequences can affect the recall performances. Indeed, using the concept of bigram, it is possible to obtain the strength of the association between the different items and assess the regularity of the sequences (*Rice and Robinson, 1975*). This regularity in the data is studied by comparing the serial position recall curves computed on sequences of low and high summed up bigram frequency, as represented in Figure 2.8.

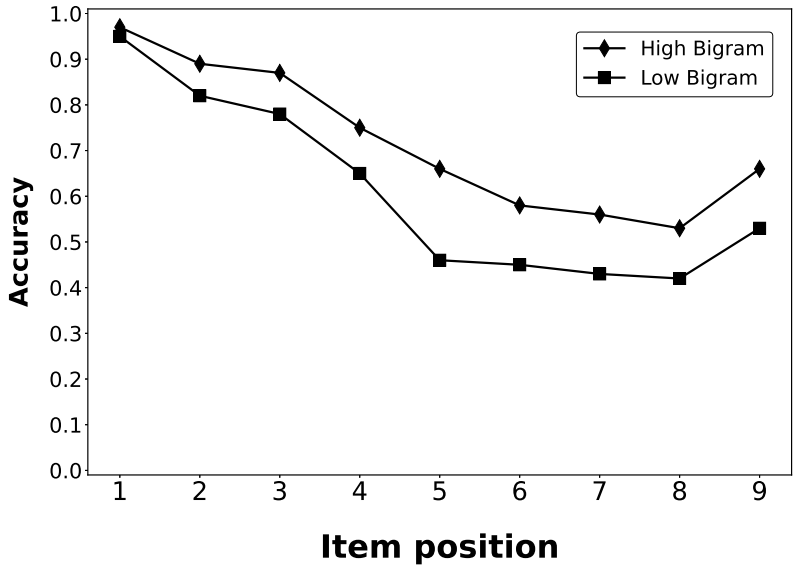


Figure 2.8: Serial position recall curves for sequences of low and high bigram, figure adapted from *Kantowitz et al., 1972*.

The two serial position recall curves clearly show the impact of structural regularities on the recall performance, with better probabilities of recall for the sequences with a high bigram frequency. These differences are more pronounced for elements from the middle of the sequences onwards. This is because, at the beginning of the recall, the primacy effect dominates, leading to high recall performance no matter the regularities present in the data. This effect is no longer present for items in the middle of the sequence. Therefore, structural regularities have a stronger impact for these positions (*Kantowitz et al., 1972*).

Chapter 3

Models of serial recall: a literature review

3.1 Importance of modeling

The brain is a complex structure involving many different mechanisms and interactions arising from information exchange between billions of neurons forming trillions of connections. Therefore, it is not easy to understand how the brain processes the information it received to achieve a desired output. This is why modeling is important as it will help to better understand its functioning.

Modeling tries to explain different observed behaviors by implementing mechanisms such that these outputs can be reproduced using the same information as the brain receives. These mechanisms are generally a simplification of the reality, but they enable a better understanding of what could happen in the brain.

However, models should always be used in combination with experimental refinements, to ensure that the model conforms as closely as possible to the experimental data. In fact, it is possible that two contradictory mechanisms are both capable of predicting observed behavior. To determine which of the two is the most plausible, new real experiments have to be constructed until one of the two mechanisms is no longer able to predict the correct outputs.

In conclusion, modeling the brain is essential to get a better understanding of its behavior, but care must be taken when interpreting the mechanisms involved.

3.2 Models of serial recall

Throughout the decades, many different models have been created to reproduce the different results observed in real serial recall experiments. These models, allowing for a better understanding of working memory, diverge by their assumptions and implemented mechanisms, but also by the effects they can reproduce. This section aims to describe the most predominant models of serial recall that have been developed.

3.2.1 Chaining model

Chaining model is the oldest and most simple model of serial recall. This model is based on the formation of associations between consecutive items of a list during the learning phase, such that, during recall, each item associatively evokes the next item in the sequence. This means that the predicted item is used as a cue for the prediction of the next item.

However, there are several objections to this very simple model. There are no mechanisms to recover from errors, as the cue for the next prediction coming from an error differs from the correct cue. Moreover,

repetitions in the sequence would lead to the same cue, which is problematic, as it will be used to predict items of the sequence (*Mewhort et al., 1994*).

To avoid these problems, more complex methods can be constructed based on the chaining principle. TODAM model only cue if the previous recalled item was correct, otherwise an approximating cue is used, and associations are done on several items instead of only the direct consecutive item. This allows to reduce the impact of single errors and to differentiate repetitions (*Murdock, 1987*).

Despite these amelioration, chaining models remain insufficient to explain some observed behaviors on confusable and non-confusable items. For lists of alternating confusable and non-confusable items, chaining models can be constructed such that they are compatible with the saw-tooth pattern of the error curves. However, the fact that the presence of confusable items in a list generally has no detectable influence on the probability of correctly recalling surrounding non-confusable items cannot be explained by the model (*Henson et al., 1996*).

The last evidence against that model comes from the frequency of relative errors. Indeed, after an error in the response, the model will produce an imperfect cue, which will increase the probability that the next response is the item that followed the erroneous response in the list. The two responses are in the correct relative order, but both are in the wrong position (*Henson et al., 1996*).

In conclusion, chaining models predict an effect of confusable items on the recall of subsequent non-confusable items, which is not observed in real data. Whether the confusable items are correctly recalled or not, it has not detectable impact on the recall of the next non-confusable item. Chaining models are thus not able to explain such behavior.

3.2.2 Primacy model

The primacy model does not rely on associations between items of the sequence as in chaining models but on a primacy gradient of activation of the items. With this approach, the model encodes the order of the items, using associations between the items and the start of the list to create the activation gradient (*Page and Norris, 1998*). The strength of associations in the gradient decreases with the item position in the list, as represented in Figure 3.1.

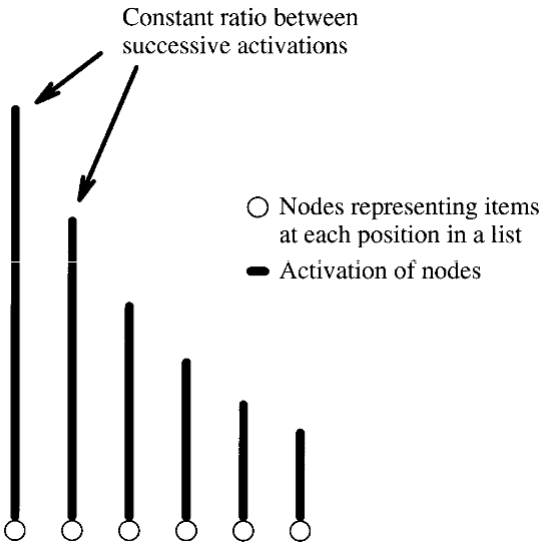


Figure 3.1: Representation of the primacy gradient, Figure from *Henson et al., 1996*.

In the recall phase, the model chooses the node with the highest activation. The chosen node is then suppressed, and the model can start to make the prediction of the next item, using the same principle. However, this model cannot be directly used; otherwise, it would lead to a perfect recall of all the sequences, since the highest activation would always correspond to the correct item.

To be able to reproduce real experimental data, an independent gaussian noise of zero mean has been introduced to each node, such that the activation of an incorrect item is now able to be higher than the activation of the correct one, allowing the model to produce some errors.

During the recall phase, all the items that have not been recalled compete together for the next prediction. At the beginning of the list, the early items are easily recalled correctly because the activation gradient is constructed with a ratio rule. The activation of early items is much larger than the other such that it will be less probable that the gaussian noise allows an other node to win. As the early items are better recalled, the model is able to reproduce the primacy effect. At the end of the recall, the decrease in the number of item nodes compensates for the lower difference between the activation of the different nodes, leading to the presence of a recency effect (*Henson et al., 1996*).

Even if this model can reproduce the typical bow-shaped recall curve, one major limitation arises from the type of errors the model can produce. In fact, since the model suppresses the node of an item once it has been chosen, no repetitions can occur in the model. Moreover, the selection of the item prediction is only based on the items present in the original sequence, not allowing any intrusion or omission errors. The model can have repetition errors, by having a slow wear-off of the response suppression, but still no omission or intrusion can occur.

Despite this major limitation, the main advantage of the primacy model is its explicit representation of the recall process, making it very easy to understand the behavior of the model and is able to reproduce different experimental effects such as primacy, recency, list length, etc.

3.2.3 SOB-CS model

SOB-CS model and the next model, TBRS*, were both introduced to model complex-span performance, but with very different mechanisms involved.

The SOB-CS model, which stands for Serial-Order-in-a-Box model of Complex Span, assumes that the distracting information is encoded exactly the same way as list items. This means that the distractors are associated to the current context. SOB-CS model can account that processing a larger number of different distractors in a fixed time window leads to worse performance with varying distractors (*Hoareau et al., 2017*).

This model is implemented using two fully connected layers: an item layer, representing the items, and the position, representing the serial position in the list. The items are vectors of -1 and +1, constructed to reflect item similarity, while positions are vectors between -1 and +1, reflecting similarity between ordinal position, exponentially decreasing with their distance. The same representation as the item representation is used to create the distractors (*Oberauer et al., 2012*).

The encoding of the items is done through Hebbian learning (*Milner, 2003*), by associating the items with their positional representation using a time-dependent process. The longer the time spent to encode the items, the higher the encoding strength. Recalling of the items is done by activating, for each position, its corresponding position pattern. This is fed to the model to compute the activation pattern of the item. To determine the item to recall, similarity measures are computed between the model output and each

candidate item. These similarities are converted into probabilities, and the recalled item is sampled from this probability distribution.

Intrusion errors are possible in this model, as all the items of the vocabulary are included as recall candidates. To avoid recalling the same item continuously, an antilearning mechanism, which is a Hebbian learning mechanism with negative encoding strength, has been implemented. This allows to suppress the activity of the item once recalled. In addition, a gaussian noise has been implemented to represent the output inference (*Oberauer et al., 2012*).

For now, this model is limited to complex span serial recall. It should thus be improved to take into account other variations of the task as free recall, probed recall, etc. Moreover, there is no explicit relation between working and long-term memory in this model, despite the fact that this relation is important in the complex span task (*Farrell et al., 2016*).

3.2.4 TBRS* model

The TBRS* model, which stands for time-based resource-sharing, is the second main type of model developed to model the complex span task. This model relies on two main assumptions: forgetting is modeled as a time-based decay, and active maintenance is used to prevent loss of information. This model is the complete opposite of the SOB-CS model by its assumptions (*Logie et al., 2021*).

When attention is focused on distractors, memory performance will decrease. The longer the time devoted to the distractors, the more important the performance decline. Inversely, performance increases when more time can be dedicated to information refreshing. These predictions can be represented by the cognitive load, the fraction of available processing time between two memory items that is taken up by the distractor task.

$$CL = \frac{aN}{T}$$

The cognitive load is proportional to the number of operations in the processing episode N and a, the time demand for each operation, while this is inversely proportional to the total time available for processing. This equation is supported by different regularities observed in the experiments (*Oberauer et al., 2012*).

The local effects of dense distractors can be captured by the TBRS* model, under the assumption that people refresh items in a cumulative manner; any interruption resets the refreshing process back to the start of the sequence, but the serial position functions predicted are considerably deviated from the empirical data (*Farrell et al., 2016*).

As for the SOB-CS model, this model is not able to represent all the different experimental benchmarks, even if the results are globally more consistent with the SOB-CS model. In addition, some critics reject this model as it is unable to capture some fundamental aspects of working memory. Finally, the presence of a decay mechanism in working memory is controversial (*Oberauer and Lewandowsky, 2014*).

3.2.5 Recurrent neural networks

The recurrent neural network model is a complex model composed of processing units, artificial neurons, interconnected together in a complex network. Each neuron has its own internal state, varying continuously depending on the inputs it receives.

This model is composed of three main layers. The input layer represents the different elements of the sequences during encoding, the output layer represents the output response of the network, and the hidden layer processes all the information received by the input layer. The main particularity of this type of model is the presence of recurrent connections in the hidden layer. With these feedback connections, the activation of each unit is affected by the activations at the previous timestep (*Botvinick and Plaut, 2006*).

This type of model learns the task by changing the weights of connections between neurons such that the network can correctly predict the items of the sequence. The network continuously adapts its internal structure until a desired level of performance is reached.

Once reached, the model can be tested on the different empirical benchmarks. This model is able to reproduce many effects as serial position recall curves, transposition gradients, confusability, etc.

Many empirical evidences have highlighted that recurrent synaptic connectivity plays a central role in supporting sustained neural activation, which is essential for working memory (*Goldman-Rakic, 1995*). This also applies to the serial recall task. In fact, neurophysiologic studies have highlighted that the encoding of items and their activation maintenance are performed through neurons in the prefrontal cortex (*Barone and Joseph, 1989*). Finally, the recurrent dynamic of the neural network can be accounted by the connections between the basal ganglia, thalamus, and cortex (*Beiser and Houk, 1998*).

One objection against this type of model is that some people argue that recurrent neural networks are mainly based on a chaining mechanism because their internal states are affected by their previous state. This would make this type of model inefficient as some experimental results reject this mechanism (*Henson et al., 1996*). However, the mechanisms involved by recurrent neural networks are much more complex. This model is able to reproduce the different results of the task composed of confusable and non-confusable items, which is not possible for models relying only on the chaining mechanism. Therefore, this objection is not relevant, and recurrent neural networks can be considered as valid models to represent the serial recall task (*Botvinick and Plaut, 2006*).

3.2.6 BUMP model

The last model, the BUMP model, which stands for bottom-up multi-scale population oscillator, focuses on effects of rhythm on the recall of the sequences. In fact, sequences that are rhythmically grouped are better recalled than ungrouped sequences. In addition of being better recalled, the grouped sequences also present changes in the pattern of order errors (*Hurlstone, 2020*).

To reproduce the serial recall task, this model uses a population of oscillators. These oscillators have the characteristic to be sensitive to local variation in its temporal structure and driven by bottom-up by auditory-verbal input. When grouping is present in the sequence, the experimental effects of the recall task are a bit different. The serial position curve is multiply-bowed, with a bow for each temporal association of the items (*Frankish, 1995*). Moreover, the number of transposition errors is reduced, more particularly, for items belonging to different temporal groups. However, another type of error, the interposition error, increases with this grouping effect. This type of error consists of interchanging the position of the temporal groups while keeping the position of the items within the group (*Hitch et al., 1996*).

The model associates each element of the sequence to the current state of a temporal context signal. This signal corresponds to the activity of a population of cells. Each cell within the population has its own activity that oscillates at a specific rate and phase in response to local changes in the speech envelope. Neurons having similar tuning respond in a coherent way to amplitude modulations on the relevant scale.

This means that oscillators with a rate close to the rate of the presentation of the items will respond strongly to the items, while oscillators with a larger rate, for example, close to the presentation rate of the sequences, are insensitive to the presentation of the items but sensitive to larger rate amplitude modification (*Hartley et al., 2016*).

During recall, the different states of the temporal signal are replayed, and the learned associations are used to predict the items. This model is in good agreement with the experimental benchmarks. However, this model is limited to short-term memory for auditory-verbal inputs.

3.3 Choice of the model

Now that the different possible models of serial recall have been described, we need to choose which model will be investigated in this work. We chose to implement a recurrent neural network for different reasons.

First, this type of model has been little studied. Indeed, the only contribution to this type of model is the Botvinick & Plaut model described above. Moreover, other attempts have been made to try to reproduce this type of model and their experimental results, but with no success. By choosing this type of model, we aim to show that recurrent neural networks are indeed capable of reproducing the experimental effects of serial recall. The second advantage of investigating a model that has not been studied a lot is that there exist many directions to explore the model, better understand it, and make it more neurally plausible.

In addition, neural networks have empirical support from neurobiological data, making them a potential more plausible model.

Finally, the implementation of Botvinick & Plaut model dates back to 2006. Over the past decade, neural networks have evolved enormously and are increasingly used in a wide range of applications. These evolutions introduced methods to make the model more efficient and, for recurrent neural networks more particularly, introduced new cell dynamics inspired by biology.

All these different reasons have led to the choice of neural networks to model the serial recall experiment.

Part II

Modeling background

Chapter 4

Recurrent neural networks

After the discussion about the different models of serial recall and the choice to use recurrent neural networks. This section will introduce the fundamental aspects of this type of model in order to understand the model developed.

4.1 Introduction

Recurrent neural networks are a specific architecture of a more global type of model, which are neural networks. Neural networks are computational tools used to solve complex problems by learning from experience to then be able to make predictions from a set of new information (*Wu and Feng, 2018*).

Neural networks are composed of neurons, the central units of the model, connected by edges, which transfer the information between different neurons. Each neuron processes the information it receives before sending it to other neurons. The strength of the connections determines how the signal is transmitted between two neurons and is modified during the learning process to adapt the model to solve the task.

The neurons are typically aggregated into different layers, making different transformations of the received signal. More particularly, the layers can be separated into three main categories: the input layer, which receives the different input information of the task, the hidden layers, which process and transform the input signal and the output layer, which makes the model prediction based on the received information (*Wang, 2003*).

This type of model was first derived from an inspiration of the fundamental units in the brain, biological neurons, which are the central processing unit of the information in the brain. These neurons have been artificially replicated and are used as the central units to process the information the model receives.

The different neurons in the brain are interconnected in a complex network, with different interactions. This corresponds to the different edges of the artificial network with their associated weight. In addition, these connections can be inhibitory or excitatory. These are represented in the model using negative and positive weights respectively.

4.2 Basic concepts of Deep Learning

To better understand how the model has been developed in the following chapter, essential notions of deep learning are introduced in this section.

4.2.1 Supervised learning

There are four main ways to use neural network models: supervised, semi-supervised, unsupervised, and reinforcement learning. These different types of learning each have their own characteristics and are used in different contexts. For modeling the serial recall task, supervised learning is the most appropriate.

In supervised learning (*Cunningham et al., 2008*), the model learns a mapping between the inputs it receives and its outputs. For this, the model makes predictions using the inputs given and compares the predictions to the true output, i.e. the desired output. The model is then updated to minimize the difference between the prediction and the true output, so that once trained, the model is able to make predictions for a new object described by its input attributes or to better understand the relationship between the inputs and the output. In neural networks, since the architecture is quite complex, the model is mostly used as a predictive model rather than to understand the input-output relationship.

Supervised learning problems can be further separated into two categories, which are classification and regression problems. The classification into one of the two categories will depend on how the problem will be modeled. In regression problems, the model has to predict continuous numerical value whereas in classification problems, the model has to predict a discrete class.

4.2.2 Loss function

The loss function is a measurement used to gauge the performance and accuracy of neural networks. This measure is essential for the learning process, as it guides how the model should learn, by quantifying the difference between the model's predictions and the actual target value in the dataset. This means that the lower the loss, the better the model. However, it cannot be used as the only measurement to characterize the model performance due to overfitting of the dataset. The loss used will depend on the type of task and can affect the model's behavior, such as being more robust against data outliers (i.e. Mean Squared Error) or prioritizing specific types of errors.

4.2.3 Softmax function

The softmax function (*Franke and Degen, 2023*) is a transform generally applied to the output layer of a classification problem. This allows to convert the raw output predictions of the model into probabilities of the different classes. For this, the exponential of each output is computed and is then normalized by dividing by the sum of all the exponentials, leading to the following formula.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

The exponential terms ensures that all the outputs are positive and the normalization that they are all comprised between 0 and 1, such that the output can then be interpreted as probabilities. This transform is particularly useful for understanding more easily the model's outputs, compared to raw outputs which can take any real values.

4.3 Architecture of recurrent neural networks

Recurrent neural networks (*Jain and Medsker, 1999*) are a subclass of neural network models with a specific architecture. The network is composed of three layers, common to most neural network models: an input layer, a hidden layer, and an output layer. However, compared to other models that have only feedforward connections, recurrent neural networks also have a recurrent connection in the hidden layer. This architecture is of crucial importance to model and process temporal or sequential data, which are

data related in time. The representation of this typical architecture of recurrent neural networks can be observed in Figure 4.1.

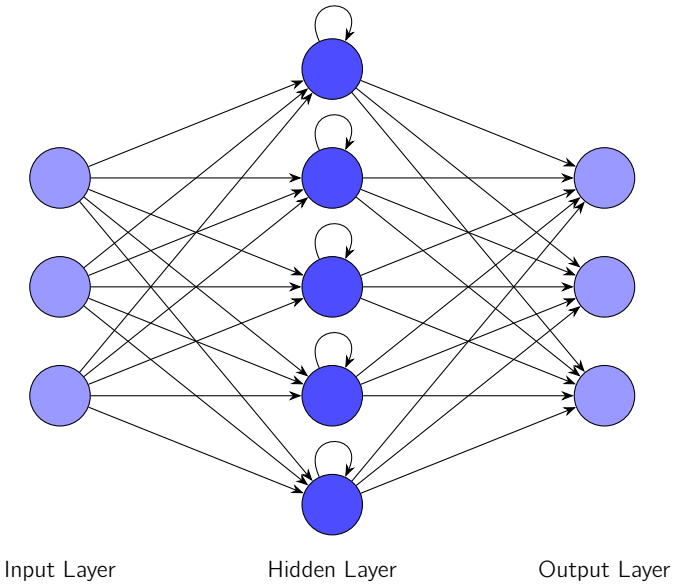


Figure 4.1: General architecture of recurrent neural networks

This recurrent connection allows the activations of hidden layer neurons at one timestep to be fed back as input to the network at the next timestep, and so have an impact on the next model predictions. Unlike other models, recurrent neural networks are then able to capture the temporal dependencies of the data and patterns present in the sequences. This type of model has been successfully applied in many applications such as speech recognition, language processing, etc.

4.4 Mathematical representation

From a mathematical point of view, recurrent neural networks can be described as two consecutive mappings: one between the input layer and the hidden layer and one between the hidden layer and the output layer. If we consider an input sequence $\mathbf{x}$ of variable length $\mathrm{T}(\mathbf{x})$,then for all t = 1,..., $\mathrm{T}(\mathbf{x})$

$$\mathbf{h}_\mathrm{t} = \phi(\mathbf{x}_\mathrm{t}, \mathbf{h}_{\mathrm{t}-1}, \theta_t)$$

The new hidden state is thus a function of the current input, the hidden state at the previous time step and the model parameters. The output prediction of the network is then computed based on the model parameters and the current hidden state value.

$$\mathbf{y}_\mathrm{t} = \psi(\mathbf{h}_\mathrm{t}, \theta_t)$$

These two mappings are the essential core of neural network models as they allow the model to make predictions. These mappings are built through the learning phase, which allows to determine the different parameters of the model.

4.5 Learning of the model

The learning phase is the most important part of the model, as it modifies the network until it can solve the task. This is done by modifying the weights of the network according to the following principle. The

network is first initialized. Then, the loss function L is computed based on the model predictions and the true outputs. This loss is used in the backpropagation algorithm to compute the changes in the network weights.

$$\Delta w_{i,j} = \frac{\partial L(\hat{y}(o), y(o))}{\partial w_{i,j}}$$

This expression computes the changes of weights required to correctly predict the data used. Once the weights changes are computed, the network weights can be adapted according to the following rule.

$$w \leftarrow w - \alpha \Delta w$$

This procedure is then repeated iteratively with the training data until the accuracy of the model for the task is sufficiently high. This learning process is also dependent of the dynamic of the cells used as it modified how the hidden states are computed.

4.6 Backpropagation in the network

4.6.1 Backpropagation algorithm

Backpropagation is an algorithm used to compute how the weights should be adjusted to reduce the difference between the desired output vector and the actual output vector in neural networks (*Rumelhart et al., 1986*). The weights are updated according to the following rule.

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \Delta \mathbf{w}$$

This algorithm is used to compute $\Delta \mathbf{w}$. These changes of weights are obtained by computing the derivative of the loss function with respect to the weights.

$$\Delta \mathbf{w} = \frac{\partial L}{\partial w_{i,j}}$$

This partial derivative is computed using chaining rule to go through all the layers of the network. The backward pass propagates the error derivatives from the output layer back toward the input.

4.6.2 Unfolding of the network

To be able to use the backpropagation algorithm on recurrent neural networks, the network needs to get rid of its temporal component. For this network unfolding is a key concept. This method converts the temporal structure of the network into a spatial one, where each timestep of the sequence is represented as a separate layer in a series. This allows to represent the transfer of information across time through the network (*Werbos, 1990*).

4.6.3 Backpropagation through time

For a recurrently connected hidden layer, the gradient-based contribution to the output error the recurrently connected hidden layer depends on two descendent layers: the output layer on the current timestep as well as the hidden layer on the subsequent one (*Sun, 2023*). Therefore, the computation of this value is given by the following expression.

$$\nabla_{h^{(t)}} L = \Big(\frac{\partial h^{(t+1)}}{\partial h^{(t)}}\Big) \nabla_{h^{(t+1)}} L + \Big(\frac{\partial o^{(t)}}{\partial h^{(t)}}\Big) \nabla_{o^{(t)}} L$$

The representation of the unfolded network, used for backpropagation is in Figure 4.2.

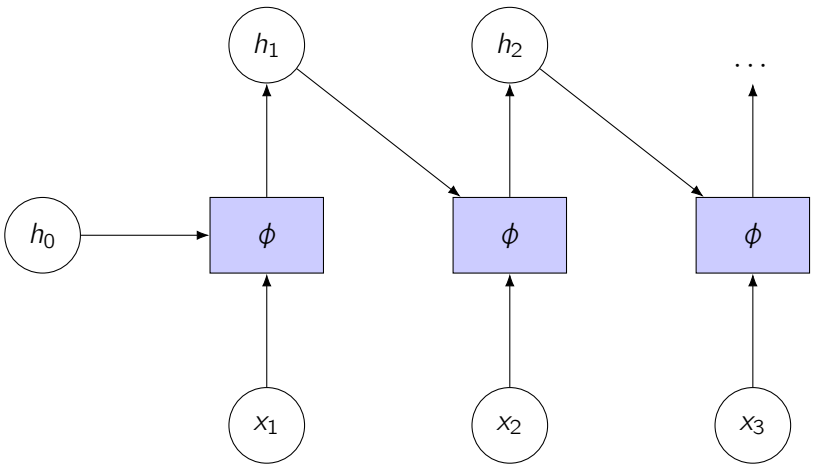


Figure 4.2: Representation of the unfolded hidden layer.

With this representation of the network, it is easy to see how the previous hidden state is used to compute the next one. With this new structure of the network, the backpropagation through the different layers of the network is now possible, allowing the use the backpropagation algorithm.

4.7 Dynamic of the cells

The second key point of recurrent neural networks is its cell dynamic, which dictates how the hidden state is processed and should evolve over the sequence. Over time, new cells have been proposed, created for computational reasons (GRU), to be closer to the brain (BRC, nBRC), etc. However, no matter the cell dynamic, the mapping used between the hidden layer and output layer will remain the same. In the following part, this section will introduce the different cell dynamics investigated with the model, explain why they have been introduced as well as their characteristics.

4.7.1 Elman cell

Elman cell is the first type of cell and has been created in the same time as recurrent neural network (*Elman, 1990*). This cell is the most simple of the ones developed as it is composed of only one processing step to compute the new hidden neurons activation values.

$$\mathbf{h}_\mathrm{t} = \sigma(\mathbf{W}_{xh}^T \mathbf{x}_t + \mathbf{W}_{hh}^T \mathbf{h}_{t-1} + \mathbf{b}_h)$$

First, the σ symbol used corresponds to a logistic regression, i.e. a sigmoidal shape function bounded between 0 and 1. For the model, a sigmoid function is used, for which the analytical expression is given by the following equation.

$$\sigma(x) = \frac{e^x}{1 + e^x}$$

With this sigmoid, the hidden neurons have a switch behavior. For small inputs, the activation of the neuron will stay close to zero and the neuron will be inactive, whereas for large inputs, the neuron activation will go to 1 and the neuron will be active. The saturation of the sigmoid function prevents to large inputs to result in too large activity that would be much larger than the activity of other neurons in the hidden layer. The sigmoid function allows a smooth transition between these two states, such that neurons can be between these two stages. The sigmoid is thus continuous, which is essential to be able to run the back-propagation algorithm on the network.

Inside this sigmoid function are present three different terms, first term is the input at the current timestep, the second is the hidden neuron activation at the previous timestep and finally a bias term. The

input is multiplied by a the matrix $\mathbf{W}_{xh}^T$, for which the dimension is given by the size of the input layer and the size of the hidden layer, such that each input node affects all the neurons in the hidden layer. In the same way, hidden layer activation vector is multiplied by the square matrix $\mathbf{W}_{hh}^T$, for which the dimension is given by the hidden layer size, such that each hidden neuron activation at the previous timestep is sent as feedback to all the different neurons of the hidden layer.

The hidden layer update equation has been formulated using vector form as all the neurons of each layers can be grouped together. This is what is down when using the network, as it allows a more efficient computation of the model.

Even using a really simple dynamic for the cell, the network is able to learn how to solve complex temporal problems, which was not possible using other types of architecture of neural networks.

4.7.2 Gated Recurrent Unit cell

The second type of cell has been introduced mainly for a computational perspective, as the Elman cells could face vanishing gradient problems. In theory, the first cell to solve this problem is the Long-Short Term Memory (LSTM) cell. However, as the Gated Recurrent Unit (GRU) cell is a simplification of the LSTM and are simpler and faster to train, only this last cell has been investigated with the model (*Salem, 2022*).

Vanishing Gradient

In recurrent neural networks, it is difficult to remember the past inputs because of the gradient descent, as the error signals flowing backward in time can vanish. This can be explained by the following expression used in the backward propagation algorithm.

$$\frac{dh_t}{dh_{t'}} = \prod_{i=t'}^{t-1} \frac{\partial h_{i+1}}{\partial h_i} = \prod_{i=t'}^{t-1} \frac{\partial f_\theta}{\partial h}(h_i, x_{i+1})$$

When the spectral radius of the Jacobian $\partial_h f_\theta$ is upper bounded by a constant strictly smaller than 1, the expression above converges exponentially to zero. This means that the contribution of past hidden states to the current loss becomes negligible, long-term dependencies are thus difficult to be learned using gradient descent learning. This problems is general to the different neural network architectures containing several layers. However, the way this problem is handled in recurrent neural networks is unique to this type of architecture (*Hochreiter, 1998*).

Description of the cell

To be able to solve this vanishing gradient problems, the Gated Recurrent Unit introduced a gating mechanism, which allows to control the flow of information in the network, by selectively remembering or forgetting the information over time. Gating is thus a mechanism, which avoid the recurrent state to go repeatedly through a squashing non-linearity, by using additive paths.

Using this gating principle, Gated Recurrent Unit cells introduced a much more complex processing of the information by the hidden layer neurons, described by the following set of equations.

$$\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{W}_z^T[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_z) \\
\mathbf{r}_t &= \sigma(\mathbf{W}_r^T[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_r) \\
\bar{\mathbf{h}}_t &= \tanh(\mathbf{W}_h^T[\mathbf{r}_t \odot \mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_h) \\
\mathbf{h}_t &= (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \bar{\mathbf{h}}_t
\end{aligned}$$

The equations introduce two different gates, a reset gate $\mathbf{r}_t$ and an update gate $\mathbf{z}_t$. At each timestep, the candidate activation vector by combining all the information received from the inputs and the previous hidden states. After this, the two gates are used to determine how much of the previous hidden state must be forgotten and how much of the new candidate vector must be incorporated in the new hidden state value.

The candidate vector uses an hyperbolic tangent as activation function, which is a symmetrical sigmoidal shaped function, bounded between -1 and 1. This means that now the hidden activation values can also take values between -1 and 1, contrary to the Elman cell, where the activation was restricted between 0 and 1. Negative activation values means that the neurons will inhibit the other neurons at the next timestep.

To better understand how the inputs and hidden state activations are processed by the cell, a block diagram is generally used to represent visually the dynamic of the cell. For Elman cell, as the dynamic remains fairly simple, understanding the dynamic is not too complicated; this is why such a diagram has not been used. However, block diagrams are more useful for the representation of more complex cells, as GRU and the others described in the following sections, as it represents all the different mechanisms involved to process the information. The block diagram of the Gated Recurrent Unit cell is represented in Figure 4.3.

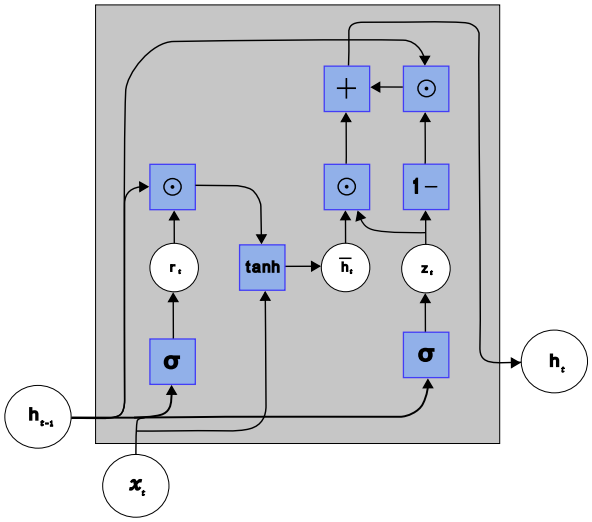


Figure 4.3: Block diagram of the GRU cell.

In conclusion, with its much more complex dynamic, Gated Recurrent Unit cells are able to handle Vanishing Gradient problems and can achieve a much faster and accurate convergence of the model.

4.7.3 Bistable Recurrent cell

Bistable Recurrent cells are a biologically inspired cell, developed to get a dynamic closer to what happens in the brain. The main characteristic of the cell is its bistable property (*Vecoven et al., 2021*).

Bistability

Bistability is the property of some non-linear dynamical system. Any non linear dynamic system can derived as the following expression.

$$\dot{x} = f(x)$$

The dynamic of such system can be studied, and more particularly its equilibrium points, which are the states of the system such that

$$\dot{x} = 0$$

These equilibrium can be stable, the system trajectories are attracted to that state and unstable ones, which repulse the network trajectories. A bistable system is a dynamical system that has two different stable equilibrium points, such that the network can converge to one of this two equilibrium depending of the current state of the system.

Bistability is a really important property as it allows the dynamical system to acquire a never-fading memory behavior. This means that, if no significant input is delivered to the system, the system will remains to one of the two stable equilibria, depending of the previous activity of the network.

Bifurcation

Bifurcation is a phenomenon appearing in non-linear dynamical systems (*Hochreiter, 1998*) . It occurs when a smooth change of a parameter value, called bifurcation parameter results to a change of the system behavior. More particularly, at the bifurcation point, the stability properties of the equilibrium points are modified.

For the Bistable Recurrent cell, a supercritical pitchfork bifurcation is implemented for the hidden state, this bifurcation is represented in Figure 4.4. With this type of bifurcation, if the bifurcation parameter is smaller than one, the neuron is in a monostable regime, with only stable equilibrium, whereas, once the bifurcation parameter becomes greater than one, the neuron switches to a bistable regime, with two stable equilibria. Between these two is an unstable state, which separates the basin of attraction the two stable ones.

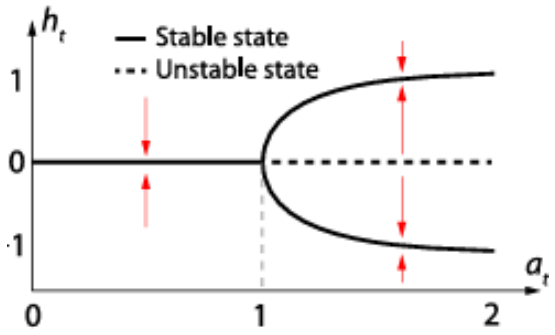


Figure 4.4: Representation of the pitchfork bifurcation diagram for the hidden state of the Bistable Recurrent cell, Figure from *Vecoven et al., 2021*.

Brain Inspiration

In the brain, neurons can also be seen as dynamical system, with the membrane potential of the neuron evolving over time, which can thus be described using differential equations, by modeling the fundamental aspects of the neuron. To implement the neuronal behavior, a control theory approach has been considered, using a recurrent feedback loop.

These biological neurons are mainly characterized by a local positive feedback, which can be reproduced in electrical circuits by using transconductance amplifiers, leading to the following expression (*Vecoven et al., 2021*).

$$I_{int} = V_{post} - \alpha \tanh(V_{post})$$

The control of the bistability property is done through the parameter α, which can range from 0 to $+\infty$. This parameter models the calcium and sodium activation channels and is used to tune the local gain of the feedback, to control the switching from monostability to bistability. This equation has been a central point to define the dynamic of the BRC cell.

Dynamic of the cell

The bistable recurrent units uses the GRU equations as a basis, and are then modified to implement the bistability and cellular memory properties of the neurons. The set of equations is described in the following.

$$\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{U}_z^T \mathbf{x}_t + \mathbf{w}_z \odot \mathbf{h}_{t-1} + \mathbf{b}_z) \\
\mathbf{r}_t &= 1 + \tanh(\mathbf{U}_r^T \mathbf{x}_t + \mathbf{w}_r \odot \mathbf{h}_{t-1} + \mathbf{b}_r) \\
\bar{\mathbf{h}}_t &= \tanh(\mathbf{U}_h^T \mathbf{x}_t + \mathbf{r}_t \odot \mathbf{h}_{t-1} + \mathbf{b}_h) \\
\mathbf{h}_t &= \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \bar{\mathbf{h}}_t
\end{aligned}$$

The first difference with the GRU equations is the the presence of the hyperbolic tangent function in the reset gate computation instead of the sigmoid function. This allows the reset gate, which correspond to the feedback parameter to take values between 0 and 2 instead of 0 and 1. This gate, corresponding to the bifurcation parameter, allows the network to switch from monostability ($r \leq 1$) to bistability (r > 1).

The second difference is in the use of the previous hidden state in the equations. Indeed, the recurrent weights are vector instead of matrices, that is why a different convention has been used compared to the GRU equation. In these equations, **U** are the weight matrices of the inputs and **w** are weight vectors of the recurrent states. Using vectors instead of matrices for recurrent connections means that, in bistable recurrent cells, there are only connections from neurons to themselves in the hidden layer, creating a cellular memory.

All the equations are represented in the block diagram in Figure 4.5.

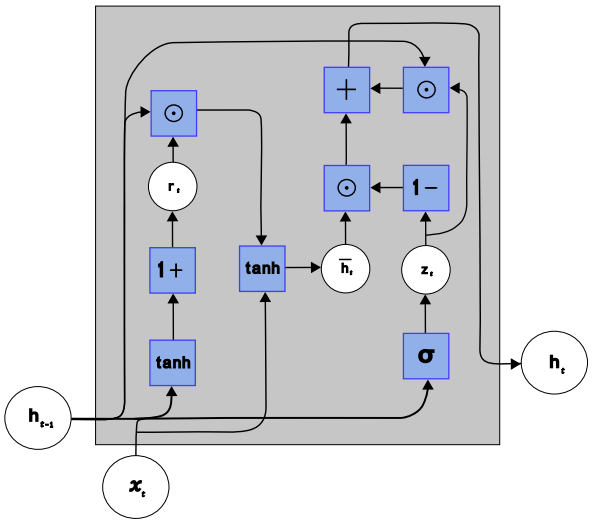


Figure 4.5: Block diagram of the BRC cell.

In conclusion, by taking inspiration from biology, a new property has been introduced in the network, bistability. With this property, the network can switch from a fading to never-fading memory and thus keep information over time.

4.7.4 Neuromodulated Bistable Recurrent cell

The last cell dynamic that has been studied with the serial recall model is the Neuromodulated Bistable Recurrent (nBRC) cell, which is an extension of the BRC introducing neuromodulation properties into the network (*Vecoven et al., 2021*).

Neuromodulation

Neuromodulation is a biological phenomenon occurring in the brain, to which all nervous systems are subjected. Neuromodulators are brain chemicals transforming the intrinsic excitability or synaptic dynamics of neurons(*Alcedo and Prahlad, 2020*). These neuromodulators have multiple actions by using different physiological mechanisms, leading to a flexibility of the neural circuits (*Nadim and Bucher, 2014*). Therefore, they play a central role in shaping the electrophysiological activity in the brain by modulating the activity of the neurons.

Dynamic of the cell

The implementation of the neuromodulation in the dynamical equations of the cell leads to the following equations.

$$
\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{U}_z^T \mathbf{x}_t + \mathbf{W}_z \mathbf{h}_{t-1} + \mathbf{b}_z) \\
\mathbf{r}_t &= 1 + \tanh(\mathbf{U}_r^T \mathbf{x}_t + \mathbf{W}_r \mathbf{h}_{t-1} + \mathbf{b}_r) \\
\bar{\mathbf{h}}_t &= \tanh(\mathbf{U}_h^T \mathbf{x}_t + \mathbf{r}_t \odot \mathbf{h}_{t-1} + \mathbf{b}_h) \\
\mathbf{h}_t &= \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \bar{\mathbf{h}}_t
\end{aligned}
$$

The bistable property introduced by the BRC is maintained and therefore the never fading cellular memory as well, by letting the parameter $\mathbf{r}_t$ to be between 0 and 2. On top of the cellular memory, is added a network memory by the introduction of the recurrent neuromodulated connections. These connections relax the cellular memory constraints as the previous hidden state of a neuron influences not only its own current state, but also all the hidden states of the neurons in the hidden layer. The neuromodulation introduced in these equations is thus a way of modulating the activity of the hidden layer neurons by the neurons of the same layer and their activity during the previous timestep. The block diagram representation of the equations can be visualized in Figure 4.6.

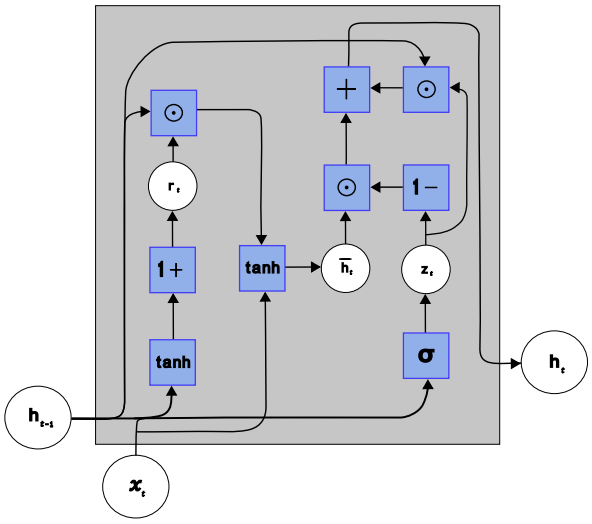


Figure 4.6: Block diagram of the nBRC cell.

The block diagram of the nBRC cell is strictly identical to the one of the BRC cell as the dynamical equations are identical, the difference between the two relies on the the structure of the weights, which are not represented in the diagram.

All the different cells introduced in this section have their own particularities, which can be for computational reasons or to get more biologically plausible dynamics. These cells will be investigated on the serial recall model and more particularly, their impact on the results of the model will be assessed, by checking if the results obtained are closer to those observed experimentally.

4.8 Hyperparameters

Hyperparameters are important parameters, sets before the training of the model, which are used to control the learning process of the model. These parameters have an impact on the model convergence, complexity and performance. It is therefore essential to get sufficiently good parameters so that the network is able to solve the presented task.

For recurrent neural networks, there are three important parameters: the batch size, the number of neurons and the learning rate. To find the optimal values of these parameters, different techniques can be used as grid search, random search, bayesian optimization, etc.

4.8.1 Batch size

The batch size is a hyperparameter of neural network models, which correspond to the number of samples of the dataset processed before updating the model parameters. This parameter has a crucial impact on the training dynamic of the network, more particularly on computation time and resources.

The choice of the batch size value corresponds to different types of gradient descent. When the batch size is equal to the size of the training samples in the dataset, batch gradient descent is used. This computes the error distance for each sample of the training set, but only update model parameters after a complete dataset pass, which results in efficient computation, but at the risk of not always achieving the most accurate model.

The second type of gradient descent is stochastic gradient descent when the batch size is set to one, meaning that the model is updated after each sample of the training set. This could result in more accurate models. However, stochastic gradient descent is not computationally cost effective as the model is updated very often and could lead to noisy gradients.

Finally, when the batch size is smaller than the dataset but greater than one, mini-batch descent is used. This method is generally a good compromise as it combines the advantages of stochastic and batch gradient descent, by dividing the dataset into smaller batches and then updating the model once a batch has been processed. This leads to a balance between the computational efficiency and the accuracy.

In conclusion, the batch size impacts when the model is updated and thus the learning process. Its value must then be chosen carefully and will depend on the application. Moreover, modifying the batch size value could imply to tune the other model's hyperparameter again.

4.8.2 Number of neurons

The second hyperparameter is the number of neurons and corresponds to the size of the hidden layer. This parameter must be sufficiently large such that there are enough neurons to process the given information

to solve the task. Increasing the number of neurons increases the complexity of the network such that it can capture more complex decision boundaries to better solve the task.

However, if the number of neurons is too large, the learning process could be slower as there are more parameters to fit, more computationally costly as the model is bigger and leads to poorer results especially if the data is noisy, as the network would have more neurons to process the information and so to capture the noise.

4.8.3 Learning rate

The last parameter of the model is the learning rate, which represents how much the model weights are updated during the backpropagation training process, as represented by the following equation.

$$w \leftarrow w - \alpha \nabla L(w)$$

On one hand, if the learning rate is too small, the model converges slowly to its solution, requiring a lot of epochs, which leads to inefficient resource use or either get stuck in a local minimal of the loss, leading to a sub-optimal solution. On the other hand, for too large learning rate, the model could oscillate around a minima or weights to overflow due to their large changes. This means that the learning rate is a trade-off between the convergence rate of the model and overshooting.

To get an alternative to a constant learning rate, learning rate scheduling can be used. This consists of decreasing the learning rate value across the training, according to a specific dynamic as linear or exponential decay. This allows to have large network changes at the beginning of the learning phase, and reach a set of weights that are good enough. With the evolution of the learning phase over the epochs, the smaller learning rate allows fine-tuning of the network weights.

In conclusion, the hyperparameters are fundamental to obtain a trainable task to the desired task and must then be chosen carefully.

Part III

An implementation of a recurrent neural network model of serial recall

Chapter 5

Description of the model

This section explains how the psychological task of serial recall has been modelized such that it can be used to train the network. Then, the specific implementation of the network is discussed as well as its training procedure. The defined implementation of the model is essential, as it will be used to reproduce the different experimental results. The global architecture is similar to the Botvinick and Plaut model (*Botvinick and Plaut, 2006*).

5.1 Definition of the task

As previously described, the serial recall task is composed of two different phases, an encoding phase, where all the items of the sequence are presented to the participants, and the recall phase, where the participants need to call back the different elements of the sequence in the same order they were presented.

As there are many variants of the serial recall task, the specific task used in this work must be defined. The task considered is immediate serial recall, implying that there is no delay between learning and recall. The sequences used are composed of letters without repetitions. The letters are considered as independent; i.e. there are no notions of similarity between the letters, familiarity, etc. This task served as the basis for the various experiments. However, it was also adapted to suit the experimental results to be reproduced.

5.1.1 Absence of repetitions

The absence of repetitions of the same item in the sequences presented is essential, as repetitions of items modify the experimental results observed. In fact, the Ranschburg effect (*Greene, 1991*) suggests that in short sequences, the presence of repeated items leads to poorer recall performance. This effect has been interpreted as the result of a guessing strategy that avoids recalling the same remembered items multiple times as possible responses.

However, this effect is controversial due to the difficulty of reproducing it experimentally. For this reason, item repetitions have not been considered in the sequences presented as model inputs. Nevertheless, even if no repetition of elements is presented to the network, this does not mean that the model cannot make repetition errors during recall.

5.2 Modelization of the task

The first step to create the model is to define how the task is represented, such that the model can be trained on it afterwards. As supervised learning is used, two different elements have to be defined, the

inputs to the network, which are the data processed by the network and used to make predictions, and the true outputs, which are the desired outputs that are compared to the model predictions to adapt the model during the training phase.

5.2.1 Representation of the task

The whole serial recall task has been represented by the model, which comprises the encoding and the recall phase. In this way, the inputs of the model are the different items of the sequence, which are presented uniquely during the encoding part. From these inputs, the model has to predict the sequence two times: once during the learning part, where each letter presented to the model is directly recalled, and once during the recall part, where the model has to recall the sequence previously presented.

As the sequences are composed of letters, they cannot be used directly to feed the network, but should first be converted into numerical values. As there are 26 letters in the alphabet, each letter will be converted into a numerical value from 1 to 26, where the value is given by the position of the letter in the alphabet.

It is also important to define the modality of presentation of the sequences to the participants, as it will change how the items can be presented to feed the network. Two main presentation modalities are used for the serial recall task, visual and auditory presentation.

For auditory presentation, sounds corresponding to the different letters are heard by the participants. It can be supposed that they directly recognize the sounds such that the sequence of items can be directly used as inputs to the network. For the visual presentation, an intermediary step could be added to the model to better represent what happens in humans. Indeed, before trying to remember the letters of the sequence, the participants should recognize them visually. This step could also be modeled, by using a convolutional neural network, to recognize the letters presented on the images. However, as this work focuses on the serial recall model, this intermediary step will not be considered, and it is assumed that the sequences are presented orally.

5.2.2 One hot encoding

One hot encoding (*Seger, 2018*) is a way to represent a variable of n states on n bits, where only one bit takes a value of one, corresponding to the state of the variable. This allows for the efficient use of categorical data in machine learning models. In fact, this method removes the ordinality, as many variables have no intrinsic order. Thus, using numerical values could be incorrectly interpreted as a ranking, leading to false predictions. Moreover, one hot encoding could lead to better performance as it could better capture complex relations of the data.

This type of representation of the data is particularly well suited for the input representation of the items of the sequence. Indeed, the items are one of the 26 letters of the alphabet and are independent from each other. This method will then be used to represent the input sequences, where each letter of the sequence is represented using one hot encoding.

5.2.3 Classification problem

The representation of the outputs can be done with two different approaches, classification and regression, which will define how the predictions of the model will be handled. As in the psychological task, the participants should for each item say a letter, a classification approach is most suitable to the task. Indeed, with this approach, for each prediction, the model has to choose one label corresponding to one of the different letters possible.

5.2.4 Recall item

Once the learning part of the sequences is completed, the recall part begins, and the network has to call back the sequence. For this, the inputs to the network should indicate that the network has reached this phase. A new token is introduced for this purpose. This token, named the recall item, is introduced as a new neuron in the input layer and will be used in the following way. First, during the learning part, the neurons corresponding to the items of the sequence are activated consecutively. Then, from the beginning of the recall part, the recall item is activated until the end of the sequence. The recall item is thus a way to indicate to the network that it has to recall the sequence, without giving any hint about the letters to recall.

5.2.5 End of the list item

In addition to recalling the presented sequence, the participants stop telling items once they have recalled the whole sequence. In the model, no mechanism indicates that the network has finished its recall. To add this mechanism, a new token has been introduced at the end of the sequence. This element is present at the end of all the sequences. It can be predicted at any time by the model and will be used by the model that it has finished to recall presented the sequence.

A main difference from the Botvinick model is that in their model, the end of the list item can only be predicted at the last element of the sequence. This means that the model can forget to predict that the sequence is finished, but cannot predict that the sequence is finished earlier than expected. However, in experimental recall of long sequences, participants may not remember all the items of the sequences, and thus finish the recall before expected; it is therefore important to allow the model to exhibit this behavior as well. For this reason, this constraint has been relaxed in this model, letting the model predict the end of the list item whenever it wants.

5.3 Network structure

The network architecture used for the model is a recurrent neural network, which uses the previous hidden states to make the prediction at the new timestep. The input and the output layers have the same dimensions, equal to 27, which are given by the number of letters plus one node for the recall or end of sequence item. For the hidden layer, 200 neurons have been used to have a sufficiently large number of neurons such that the network can learn the task. The learning rate has been set to 0.001, as this is the standard value for neural network training.

5.3.1 Output feedback

However, for the model to correctly recreate transposition errors, the model architecture cannot be used directly and must be modified. In addition to the hidden layer feedback, an output feedback has been added, which uses the previous model output to compute the new hidden state. This type of feedback is usually used in auto-regressive models, to generate elements using last model prediction such as in text and music generation. The addition of this output feedback will modify the dynamical equation of the hidden state, which is now given by the following expression.

$$\mathbf{h}_\mathrm{t} = \sigma(\mathbf{W}_{xh}^T \mathbf{x}_t + \mathbf{W}_{hh}^T \mathbf{h}_{t-1} + \mathbf{W}_{oh}^T \mathbf{o}_{t-1} + \mathbf{b}_h)$$

It is important to define what will be used as the output feedback. Three different possibilities could be considered: the direct activation of the output layer, the probability of the different letters, or the letter predicted by the model.

From the point of view of the real experiments, the participants use their last predictions as a hint to make their next prediction. Indeed, all the letters of the sequence that the participant has learned are linked by their position in the sequence and therefore the next item recalled is conditioned by item recalled just before. For this reason, the output feedback used in the model will be the last prediction of the model, regardless of whether the prediction was correct or not.

Now that the output feedback has been introduced, the complete architecture of the network can be described. This model uses a classical recurrent neural network architecture, where a softmax function has been applied to the activation of the hidden neurons such that the model output is the probability distribution of the letter. An output feedback has also been added such that the last prediction of the model is used to compute the new state of the network. This modified architecture is represented in Figure 5.1.

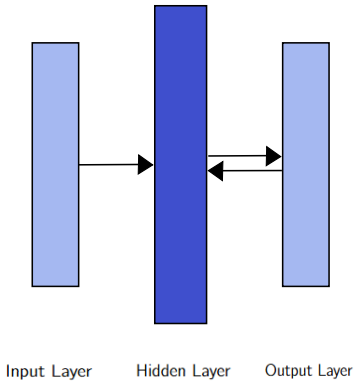


Figure 5.1: Schematic of the complete architecture of the serial recall model, with the output feedback.

The output feedback is essential for the model to produce transposition errors, and its impact on the experimental results will be discussed more deeply in the chapter on the reproduction of experimental results.

5.3.2 Network initialization

Before starting the training of the network, all the different weights have to be initialized. For the whole network, a Xavier uniform initialization has been used (Glorot and Bengio, 2010). This method consists of initializing all the biases to zero and the weights according to a uniform distribution.

$$W_{ij} \sim \mathcal{U}\Big[-\frac{\sqrt{6}}{\sqrt{fan_{in}+fan_{out}}}, \frac{\sqrt{6}}{\sqrt{fan_{in}+fan_{out}}}\Big]$$

fan_{in} is the size of the previous layer, i.e. the number of columns in matrix $\mathbf{W}$, and fan_{out} is the size of the current layer, i.e. the number of lines in matrix $\mathbf{W}$.

The idea of this type of initialization is to define the initial weights of the network such that the gradients and activations can flow effectively during both forward and backpropagation passes. The scale of the random initialization is chosen by considering the number of input and output units of the different layers of the network to determine the bounds of the uniform distribution.

5.3.3 Model Predictions

The most crucial part of the model is to determine how the model's outputs can be used at each timestep to make predictions on the letters to recall. First, the softmax function is applied to the output layer,

which leads to a probability distribution of the different letters.

With this probability distribution, the model still needs to select the letter it will predict. For this, a deterministic approach has been chosen, which consists of predicting the letter with the highest probability. The relevance of this approach will be discussed later, based on the model outputs.

5.4 Training

To train the model, sequences composed of one to nine items are presented cyclically to the network. This means that the model will use a sequence of one item to do the first update of the model, then two items, and so on until nine items. Once the sequence of length nine has been processed, the sequence length comes back to one and continues to change with the same dynamic. This change in sequence length is essential to obtain the correct experimental results. Indeed, the results of the model trained on only one sequence length are totally different and do not correspond to the experimental ones.

The second fundamental characteristic of the training is also related to the dataset used. In fact, the dataset used for the training is not fixed. This means that all the sequences presented to the network in the training are different. This is possible as there exists a really large number of sequences that can be formed with the 26 letters of the alphabet. Indeed, the number of sequences of length i with no repetition and where the order is important can be computed using combinatorial analysis. Therefore, the total number of sequences of lengths from one to nine is given by the following expression.

$$\text{Total number of sequences} = \sum_{i=1}^{9} A_{26}^{i} = \sum_{i=1}^{9} \frac{26!}{(26-i)!} = 1,2\ 10^{12}$$

With such an extremely high number of sequences, it is possible to create a dataset that is not fixed. However, when creating the dataset, it is not excluded that the same sequence will be created multiple times. For sequences of very small length, this will happen quite regularly, as, for the most extreme case of possible sequence of length 1, only 26 different sequences can be created, i.e. each sequence corresponds to one letter. This will happen very less frequently for longer sequences like, for example, sequences of length five, where the total number of sequences is given by:

$$\text{Total number of sequences of length five} = \frac{26!}{(26-5)!} = 7,893\ 10^{6}$$

The total number of sequences of five items is already very high. However, this is increasingly larger for sequences composed of more items, as this is shown for sequences of nine items.

$$\text{Total number of sequences of length nine} = \frac{26!}{(26-9)!} = 1.133\ 10^{12}$$

5.4.1 Padding the sequences

As discussed previously, the dataset is composed of sequences of different lengths. Therefore, the sequences that will be fed into the network will have different sizes. However, the implementation of the dataset requires that all the sequences have the same dimensions to be handled correctly. A first solution is to create all the sequences one by one each time it is needed, but this would be very computationally heavy. The second option, and the one that will be used for the data, is to use padding.

Padding (*Hashemi, 2019*) is a technique that consists of adding special tokens to the sequences to make them uniform in length. For this, post-padding, which consists of adding the tokens at the end of the sequences, has been used for both the inputs and outputs until all the sequences have the length of

the longest one.

For the inputs, as a value of one corresponds to an activity in the node and a value of zero no activity, the padding tokens added will be zeros such that they will not generate activity in the network. For the true outputs, as the letters plus end of list item are converted into integers from 0 to 26, the padding will add values equal to -1, which allows to distinguish them from the real elements of the sequences. Once this method is applied to the dataset, the network can handle correctly the sequences it receives.

5.4.2 Batch size

As explained before, the batch size will define how many inputs will be processed by the network before updating the model. To choose this value, an analogy with the brain has been considered. The brain has the ability to capture each new information, process it, and adapt its behavior based on what it received. This means that the brain is very flexible and constantly adapts to its environment. This property will be reproduced with the model. For this reason, a batch size of one has been used, meaning that the model will be updated after each sequence the network receives, reproducing the ability of the brain to adapt to each new information.

5.4.3 Loss

To define the discrepancy between the predicted sequences and the true sequences, the loss function has to be defined, such that it can be used to adapt the model. There are two main loss for classification problems, the cross entropy and the Kullback Leibler divergence. While both losses can be used and would give very similar results, the Kullback Leibler divergence will be used as it better corresponds to the description of the output layer of the network, which is a probability distribution (*Shlens, 2014*).

Indeed, since the activity of the neurons in the output layer is passed through a softmax function, the output can be interpreted as the discrete probability distribution of the different letters. On the other hand, the true output can also be interpreted as a probability distribution, where the true letter has a probability of 100% and the others a probability of 0%.

Expression

The Kullback Leibler (KL) is a measure of the divergence between a probability distribution and a baseline distribution. The smaller this loss, the closer the two distributions. This loss can be interpreted as how much information is lost when the computed probability distribution is used instead of the real one. The expression of the loss between the predicted output $\hat{y}$ and the true output y is given by the following expression.

$$L(\hat{y}, y) = y \ \log \frac{y}{\hat{y}} = y \ (\log y - \log \hat{y})$$

This implementation of the loss requires that both arguments are given in the log-space. Moreover, the loss is averaged over the observations for each given batch. As a batch size of one is used in this model, the mean reduction has strictly no impact on the training.

Removal of padding values

It is important to remind that, as the sequences of the dataset have different lengths, padding tokens have been introduced. It is important not to consider them when computing the loss function; otherwise, artificial items would be used to update the model.

There are no inherent mechanisms to the KL divergence to not take the padding into account. As all the padding tokens in the output sequences are elements equal to -1, a mask has been created to solve this problem. A mask can be seen as a filter that blocks some parts of the inputs or outputs such that the network can only treat the important information. In this model, the mask will block all the items equal to -1, such that the loss sees the true length of the sequences.

5.4.4 Teacher forcing

A last mechanism that was introduced in the Botvinick model is teacher forcing. This is an algorithm used to train the model, where the ground-truth item is fed back as the output feedback instead of the letter predicted by the model. This is analogous to hearing oneself emit an item, which has a triggering effect on the model's subsequent outputs.

However, the use of this mechanism to train the model and its impact on the experimental results will be discussed, and therefore, if teacher forcing is essential to obtain some experimental results or if this leads to poorer results.

5.5 Testing and stopping the learning

Now that all the details regarding the model training have been discussed, it is important to define when the model training has to be stopped. For this, a fixed dataset of 10000 sequences of composed of only six items has been created. The large size of this dataset is necessary to obtain a stable and accurate representation of the model performance. The accuracy of the model is computed every 10 000 sequences on this dataset to capture the evolution of the model training.

The accuracy of this test dataset will be used to stop the training of the network. As in the psychological experiments, participants correctly recall approximately 58% of the presented sequences of length six (*Henson et al., 1996*), the training of the network will be stopped when the accuracy of the test set achieves such performance, to be in accordance with the experimental data.

5.6 Differences with classical deep learning approach

This model uses the tools of deep learning neural networks to represent the psychological task of serial recall. However, the model used exhibits two major differences compared to a classical deep learning approach.

The first difference concerns the dataset. In standard applications, a fixed dataset is used to train the model by going through the data multiple times until the model has learned the task. With this approach, the model obtained can be prone to overfitting. Overfitting means that the model fits too closely to the data used for training but is unable to make predictions on new data, i.e. the model is unable to generalize.

In the developed serial recall model, the dataset is not fixed. The model is mostly trained with new unique sequences. This is important to avoid over-learning. The model should not learn specific sequences, but rather how to recall any sequence instead. The model should therefore have a very high generalization and a low overfitting, which is ensured by this not fixed dataset. Indeed, if 100 000 sequences of each length are used to train the model, then the percentage of sequences used among all the possible sequences of the different lengths is given by the following expression.

$$\text{Percentage of sequences used} = \frac{9\ .100000}{\sum_{i=1}^{9} A_{26}^{i}} = \frac{9\ .100000}{\sum_{i=1}^{9} \frac{26!}{(26-i)!}} = 7.75\ 10^{-5}\ \%$$

This ensures that a very small part of the possible sequences have been used for the training, and therefore the model should have a good generalization, i.e. small overfitting compared to a fixed size dataset.

The second difference concerns the stopping criterion of the model's learning. In fact, in standard application, the model is trained on the task such that it can solve it as best as it can. The accuracy that the model wants to reach is the highest value as possible, and therefore the training is not stopped before the model reaches a saturation level.

This is not the case for the serial recall model. It is not desirable to obtain the highest possible accuracy value, as the model would remember all the different sequences almost perfectly and would strongly diverge to the data observed experimentally. This is why the training is not stopped when the model reaches the saturation, but when an accuracy close to the experimental results is reached, allowing to have a model as close as possible to reality.

A last point that could be noticed is that in most of the standard deep learning applications, the batch size is generally greater than one, as it ensures stability of the model learning and a more computationally effective training. This is not the case for the serial recall model; the model is updated after each sequence for the reasons explained previously.

In conclusion, despite that the serial recall model uses the standard architecture used in classical deep learning, the model has been adapted to be as close as possible to what happens for the participants, such that the model becomes a human-inspired model.

Chapter 6

Reproduction of experimental effects

With the model described in the previous chapter, it is possible to try to reproduce the different effects observed in a serial recall task. This is what will be investigated in this chapter.

This reproduction of the experimental results is a first contribution of this work as many researchers did not succeed in reproducing predictions close to the empirical benchmarks using such type of model.

The first simulation tries to reproduce the three most important effects of the serial recall task, which are the evolution of working memory performance with the length of the sequence, the serial position recall curve and the transposition gradients. For this, the model has been trained as explained in the previous chapter, using sequences of one to nine items until the model reaches 58% accuracy on sequences of length six. The model used 55556 cycles to reach the required level of accuracy. This corresponds to exposing the network to 4.16 10^{-5} % of all the possible sequences and to 0.03% of the sequences of length six. This shows that the probability that the network uses two times the same sequence of six items to train the model is very small, the model is therefore not subject to overfitting.

An other important point is that the model developed in this work requires more than two times less sequences than the Botvinick model to be trained, even if the two are similar. The potential explanations for this behavior are that the model developed here uses a better initialization of the network and that the advances in deep learning over the years could lead to a better optimization of the network.

Using fewer sequences to train the model is beneficial as it better corresponds to reality. Indeed, humans do not really learn how to do the task by being trained with a large number of sequences before being able to perform it. However, the human brain is already composed of a large number of complex connections, enabling it to process ad adapt to new information or learn new tasks very easily,as the brain exhibits a high plasticity. In the serial recall model, the model starts with a random network. The network is composed of complex connections, but these are not able to process new information, that is why the network has to be trained first with some sequences. A smaller number of sequences to train this model could be interpreted as a higher plasticity of the model, as in the brain.

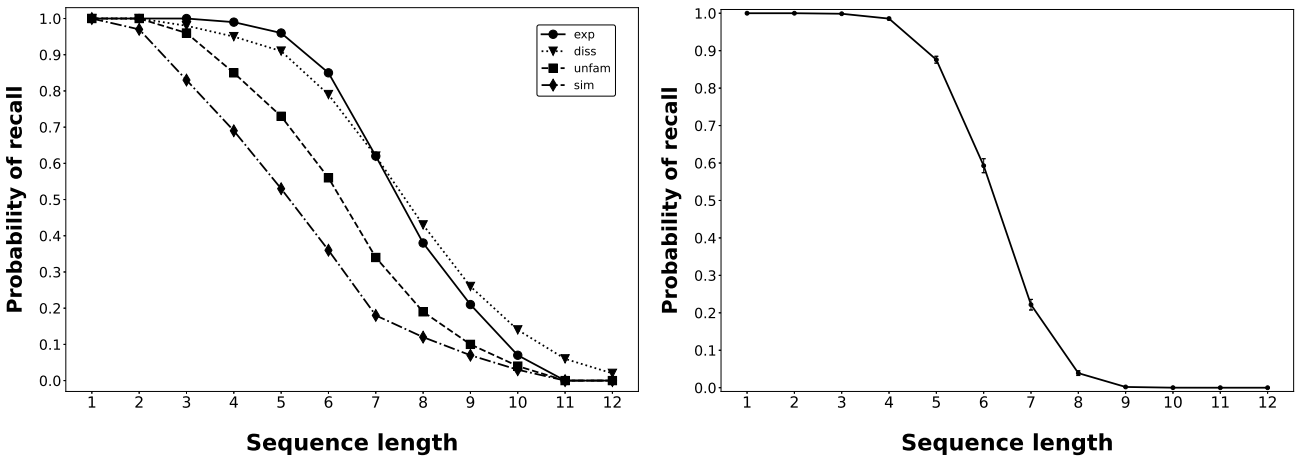
The different curves that will be presented in this chapter are mean curves, i.e. they are created by computing the accuracy on the model on 10000 sequences. This will result in averaged curves with a very small standard deviation. However, in a real psychological task, variability is a central characteristic of the curves. Consequently, it is important to try to recreate it with the model. This will be investigated in a later chapter.

6.1 Evolution of working memory perfomance with the sequence length

The first result produced by the model that will be discussed is the sequence-length curve, i.e. the percentage of correct recall of the sequences as a function of the list length. To reproduce this curve, the trained model has been used to predict 10000 sequences, for each list length. The probability of recall for a specific list length is computed as the number of sequences correctly recalled by the model among all the sequences presented.

$$P(recall) = \frac{Number\ of\ correctly\ recalled\ sequences}{Total\ number\ of\ sequences}$$

The sequence length curve computed for lists varying from 1 to 12 elements is represented in Figure 6.1.



(a) Experimental data

(b) Model predictions

Figure 6.1: Evolution of the working memory performance with the length of the sequence. The x-axis indicates the length of the sequence, while the y-axis indicates the probability that the a sequence of this length will be correctly recalled.

The prediction of the model is able to reproduce the global shape of the experimental curve, which is a decrease of the recall probability with the sequence length. This decrease has a sigmoid function shape as in real experiments.

For sequences of small length, the model is able to predict almost perfectly all the sequences that presented to it. Two different reasons could explain this behavior. The most important one is that a small number of information is presented to the model and so the network must remember only a few items. This model is thus able to retain all the information and recall them correctly, leading to high recall probability.

A second possible explanation is that for very small sequence length, the number of sequences that can be composed with the alphabet letters is limited. This means that most of the sequences has been used at least once to train the model and would overlearn the small sequences.

As approximately 55 555 sequences of each length are used for the training, all the sequences of length smaller or equal to three will be used at least once during the training to update the model as, for a fixed length, the total number of possible sequences is smaller than the number of sequences presented during the model training.

$$\text{Total number of sequences of length three} = \frac{26!}{(26-3)!} = 15600 < 55555$$

Two observations are against this second explanation. The first one is that sequences of length four and five have really high recall performance, almost identical to the smaller list lengths, even if a much smaller proportion of the possible sequences are used during model training, i.e. $15,5\%$ and $0,7\%$ of all the possible sequences of length 4 and 5 are used for model training. The second reason against this explanation is that, during the training of the model, the length of the sequences presented to the network is varied cyclically and the model is updated after the presentation of each sequence. This means that the different list lengths are all presented equivalently, there is no bias in the training, i.e. the training is not more trained on smaller sequence lengths.

For intermediary sequence lengths occur the transition region, where the model goes from a really high recall probability to a value that goes towards 0. The model is in the memory span region, it arrives at the maximum number of items it can remember properly and saturates. This is why there is this sharp transition, for smaller list length, the model can remember and recall all the items, while this happens much less frequently for larger sequences as the model achieves its memory saturation. The recall probability of sequences of length 6 is approximately equal to 58%. This was expected as the model training is stopped when the model can correctly recall 58% of the sequences of six items. This also indicates that the length of six is the memory span of the model, as it is the longest list length with more than 50% of the sequences correctly recalled.

Finally, the probability of recall of long sequences is very small, almost equal to zero. As explained above, the model has exceeded its memory span and cannot correctly recall and recall all the items in the sequence. A small difference with the experimental curves is that the probability of recall in the real experiments are a bit higher for list length of 9 and 10. This difference remains however small.

A last point that could be discussed is the generalization of the model. The model has been trained on sequences from one to nine, but has been tested on sequences length up to 12 to create the sequence length curve. This means that the model has been tested on sequence lengths that have not been used to train it. By looking at the probability of recall for the sequence length from 10 to 12, it can be observed that the three have a recall probability exactly equal to zero. This means that the model is not able to recall any sequence of these lengths from the ten thousand ones presented correctly. Different explanations could be used to highlight this behavior.

A first explanation is that the model might not be able to generalize and thus make correct predictions with sequence lengths not used to train the model. To test this hypothesis, the model was trained using the same procedure, except that sequences from one to twelve have been presented to the network. This did not make any difference during the model training and had no impact on the overall shape of the working memory performance curve. When looking at the recall probability for lengths from ten to twelve, these probabilities are still exactly equal to zero. This tells that this is not a problem of the model generalization, as no matter if the sequence lengths were used or not to train the model, the recall probability remains equal to zero.

A second explanation is more computational and related to a problem present in Elman networks processing very long sequences, vanishing gradient. Indeed, this problem is very common in such networks, making it very difficult for all the information in the sequence to pass through the network and be processed. This explanation will be investigated when comparing the different cell dynamics as the GRU cell has been specifically created to solve this vanishing-gradient problem.

The last explanation is that this recall probability of zero is just an inherent memory limitation of the model. It has been observed that the memory span of the model is equal to six, this means that the model has more difficulties to recall the larger sequences. As a result, for increasingly long sequences, the model finds it more and more difficult to recall the sequences correctly, and after a certain point is no longer able to recall any of them correctly.

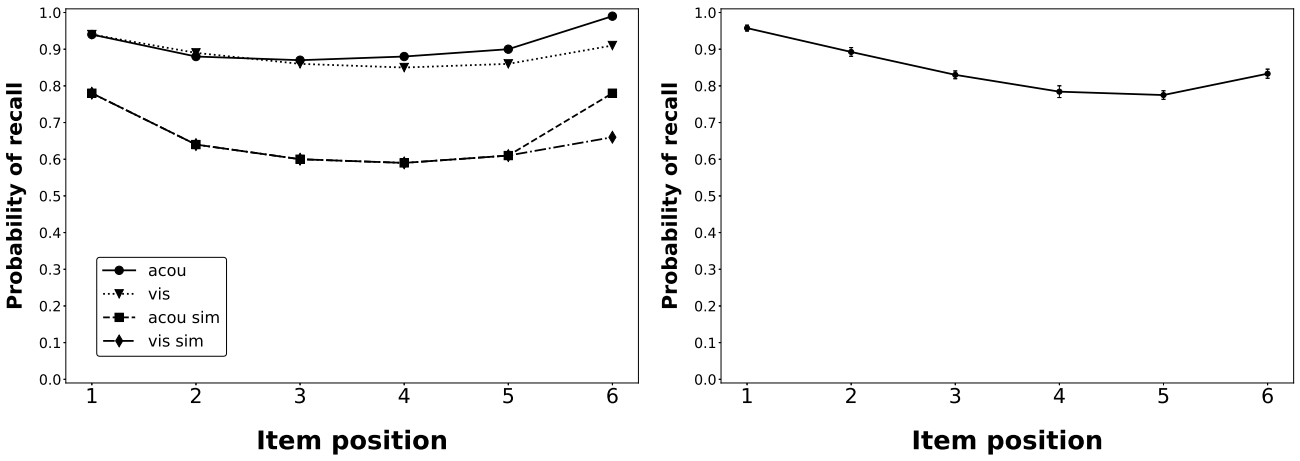
In conclusion, the serial recall model is able to reproduce quite accurately the experimental effect of the evolution of working memory performance with the length of the sequences, with a small divergence for long sequences. This curve is not the result of a bias in the model training, but is instead inherent to the model memory.

6.2 Serial position recall curve

The second effect investigated with the model is the serial position recall curve. To obtain this type of curve, the same trained model is used on 10000 sequences of fixed length to compute the recall probability of the item for each position in the sequence. This probability of recall is then computed using the following expression.

$$P(recall\ at\ position\ i) = \frac{Number\ of\ correctly\ recalled\ items\ at\ position\ i}{Total\ number\ of\ sequences}$$

As most of the serial position recall curves in the literature are done on sequence of length six, the curves predicted by the model will also be done on this sequence length. As a reminder, no notion of similarity has been introduced for now in the model, so the curves of the model relates to the recall curves for non-confusable items. The recall curve for the sequence of length six is represented in Figure 6.2.



(a) Experimental data

(b) Model predictions

Figure 6.2: Serial position recall curve predicted by the model for sequence of length 6. The x-axis indicates position within the sequence, while the y-axis indicates the probability to recall correctly the item at this position.

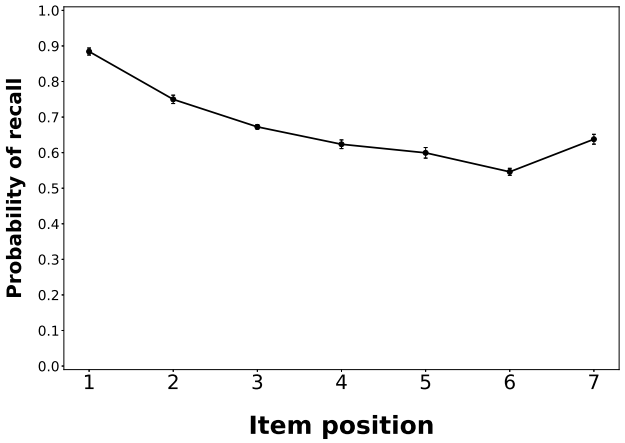
The serial position recall curve of the model has the typical bow shape observed in experiments. This means that the model better recalls the items present at the beginning and at the end of the list. Primacy and recency effects are therefore both features of this model.

When looking at the recall probabilities, it can be observed that these stay quite high for all the item positions in the list, they do not go under 75%. This makes sense since the curve of evolution of working memory performance with sequence length showed that the model's memory span is equal to six, and therefore, for this sequence length, the model must be able to remember all elements of the sequence more than 50% of the time. For this to happen, all the recall probabilities must remain high enough.

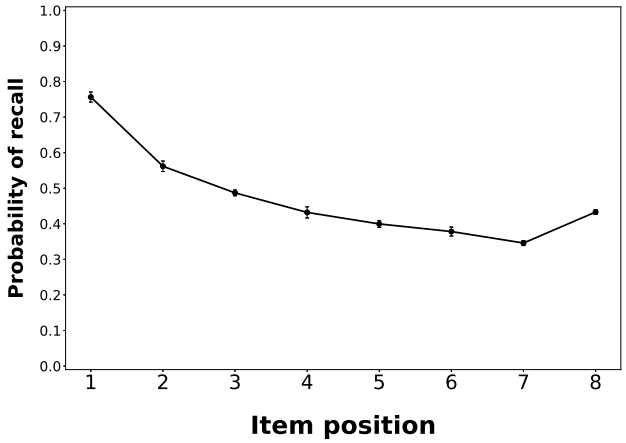
The symmetrical shape of the recall curve is also a consequence of the length of the sequence. Indeed, as this length remains quite small, the primacy effect dominates for the first elements of the sequence, leading to better performances. The recall probability then starts to decrease with the position in the sequence. However, the end of the sequence is rapidly reached, such that the recency effect starts to dominate before the appearance of a too important decrease of the recall probability. As it will be seen, the shapes of recall curves for longer sequences are quite different.

6.2.1 Recall curve of longer sequences

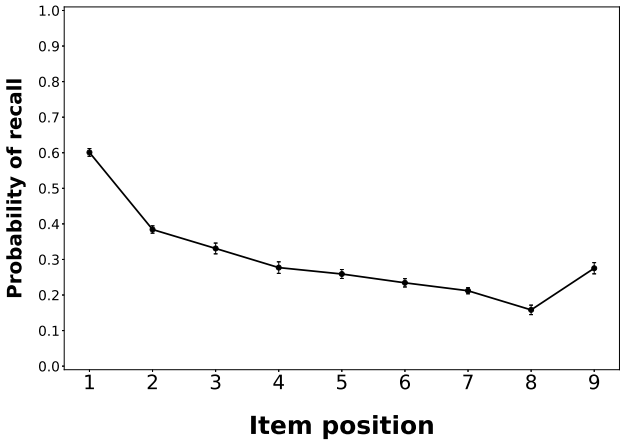
Before trying to understand how this model can exhibit this type of recall curve, it is essential to investigate the behavior of the model on longer sequences and assess if the recall curves also match the experimental data. For this, the curves for sequences of length seven, eight, and nine are represented in Figure 6.3.



(a) Length of seven



(b) Length of eight



(c) Length of nine

Figure 6.3: Recall curve predicted by the model for sequence of length seven, eight and nine. The x-axis indicates position within the sequence, while the y-axis indicates the probability that the item at this position will be correctly recalled.

The serial position recall curves obtained have all the same global behavior, but are more different from the one obtained with a sequence length of six. At the beginning of the list, the primacy effect dominates. The model is able to predict the correct item with a high probability. As the position in the sequence increases, the recall probability decreases. The longer the sequence, the smaller the recall probability for items close to the end of the list. Finally, a small increase in recall performance appears for the last item of the sequence due to the recency effect.

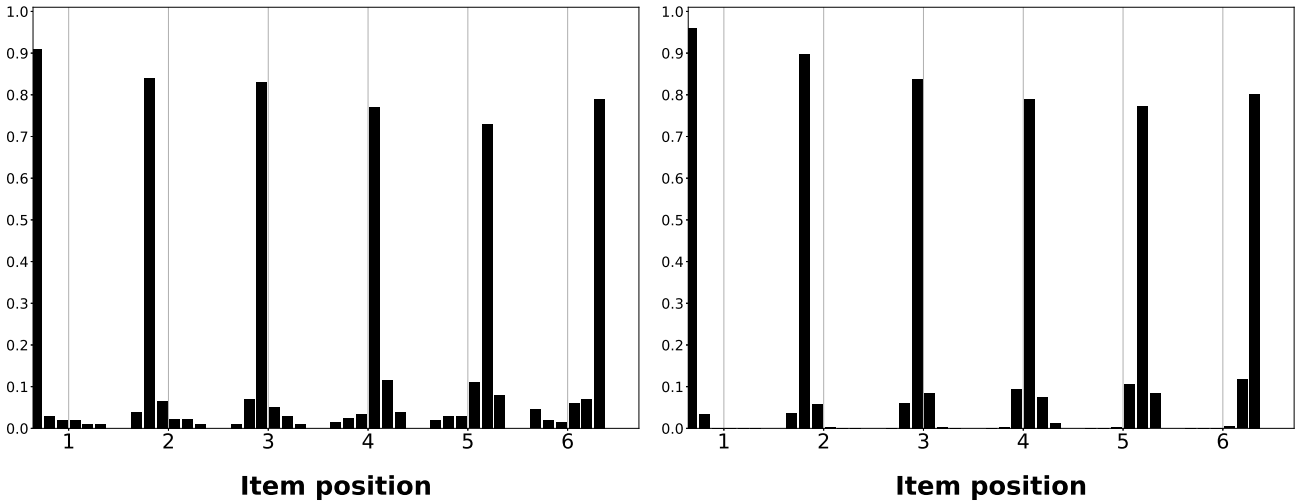
This explains why the recall probability of long lists are so small compared to short ones. As the length of the sequences increases, it becomes more and more difficult to correctly recall the items at the end of the list, and therefore it becomes even more difficult to correctly recall the entire sequence.

It is important to try and understand how the model manages to create this arc-shaped curve, typical of experimental results, and more particularly the recency and primacy effect. For the primacy effect, the model better predicts the first elements of the sequences, as they are more strongly encoded in the network because they are the firsts to be presented to the network. For the recency effect, this increase in the recall performance for the last items is mainly coming from the end of the list item. As for all the sequences, the model has to predict this particular item once the recall is finished, and as the output feedback plays an important role to predict the next item, the model needs to better recall the last item of the sequences compared to the previous ones to correctly predict the end of the list item.

In conclusion, the model is able to represent the characteristic curve of the serial recall task, for the different sequence lengths.

6.3 Transposition gradients

The transposition gradient is the last major serial recall effects and is the main source of error in the serial recall task. To measure the distribution of these errors, the distance between the prediction of the model and the true item is calculated for all the positions of the items. This results in a gradient for each position within the list. The results obtained for sequences of six items are represented in Figure 6.4.



(a) Experimental data

(b) Model predictions

Figure 6.4: Transposition gradients. The x-axis indicates the position in the target list, the clusters represent the position of this item in the predicted list.

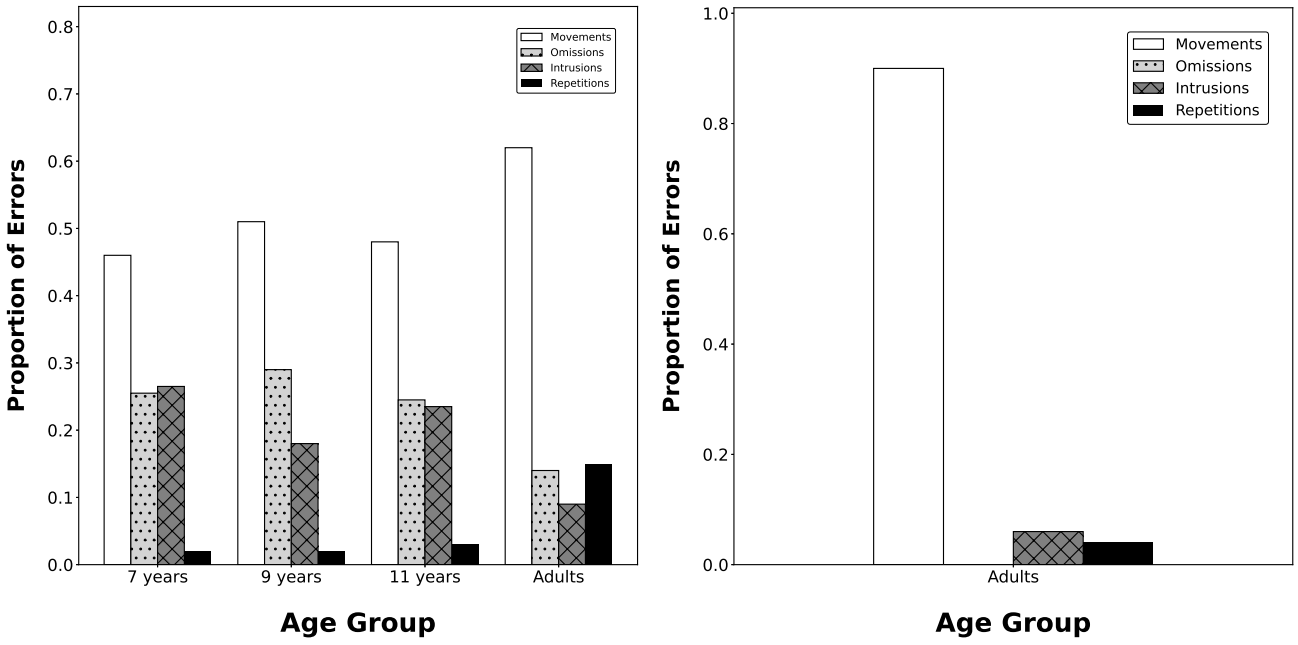
The transposition gradient obtained has the same global shape as the empirical benchmarks, with more transposition in the middle of the sequence, as these items do not benefit from the recency and primacy effects. The proportions of transposition errors with a distance greater than one are smaller than in experiments. This means that the model has a much stronger tendency to move items by a single position in the sequence when it makes a transposition error, rather than transposing them with a large distance.

A possible explanation for this behavior is that the new hidden states of the neurons strongly rely on the output feedback of the network, which corresponds to the last item predicted by the model. If the model makes a mistake, it is directly taken into account by the model, while errors made longer before are much less taken into account to make its predictions.

6.4 Type of errors

During the recall phase, participants can perform many different types of error: transposition, repetition, intrusion, and omission, which occur at different frequencies. It can be assessed if the proportions of error produced by the model are the same as the ones observed in the real task.

To compute these errors, we run through the various predicted items of the sequences. For a given position in the sequence, if the predicted item is different from the real item, then the type of error of this item is identified. If the item is in another position in the sequence, this is a transposition error. If the item is present multiple times in the predicted sequence, then this is a repetition. If nothing is predicted, it is an omission error. Otherwise, this is an intrusion error. Once calculated on 10000 sequences, the different proportions of errors produced by the model can be defined and are represented in the Figure 6.5.



(a) Experimental data

(b) Model predictions

Figure 6.5: Proportion of the different types of error, for model trained with the omission mechanism.

Many observations can be made on this result. First, the model does not produce errors of omission. This was expected as, for each timestep, the model makes a prediction by choosing between one of the output labels, i.e. the different letters and the end of list item. There is no mechanism for the model to

produce no predictions, so no errors of omission can be made. Therefore, a new mechanism has to be introduced in the model to better represent the experimental results.

By looking at the proportion of errors, it can be seen that the majority of errors are transpositions and there are too few repetition errors, compared to the experimental data. The model using Elman cells is not able to reproduce the correct proportion of the different errors present in the real psychological experiments.

Two different methods will be studied later in this work to obtain results closer to the experimental data. First, the necessity of using the teacher forcing mechanism to train the model and its impact on all the results of the model will be investigated at the end of this chapter. The second will be to investigate if the other cell dynamics will have an impact on the proportion of errors of the model.

6.4.1 Introduction of the omission mechanism

As discussed, it is necessary to introduce into the model a mechanism for not predicting any element, in order to obtain errors of omission. This mechanism is introduced at the level of the model outputs, which corresponds to the discrete probability distribution of the letters. Before the model chooses the maximum probability to predict the letter, this maximum probability is compared to a fixed threshold. If the probability is higher, then the model can predict the letter, otherwise the model does not make any prediction.

This simple mechanism is very efficient to obtain omission errors and has to be used during both model training and testing, in order to not include a bias in the results of the model's predictions. Indeed, using the mechanism only after training would lead to different results, as the model is not trained with it.

The value of the threshold is an important parameter as it will define how confident the model should be in its prediction to not make an omission. The choice of the value of this parameter was dictated by two reasons:

- How the threshold value makes it possible to obtain a proportion of omission errors close to the proportion observed in the experiments.
- How much this value makes sense from a psychological perspective.

The first point will be assessed by going through different threshold values, which are used to train the model. The proportion of omissions is then computed, using the trained model with the specific threshold and compared to the experimental data to determine a good range for the parameter value. This procedure allowed to find a good threshold value of around 50%.

The second point is to check whether the threshold value used makes sense. Indeed, for example, the threshold probability should not be lower that one divided by the total number of output classes, which corresponds to the smallest maximum probability the model could predict and would correspond to a situation where the probability of the different letters are uniform. If the threshold is below this lower bound, then the model will never produces omission errors.

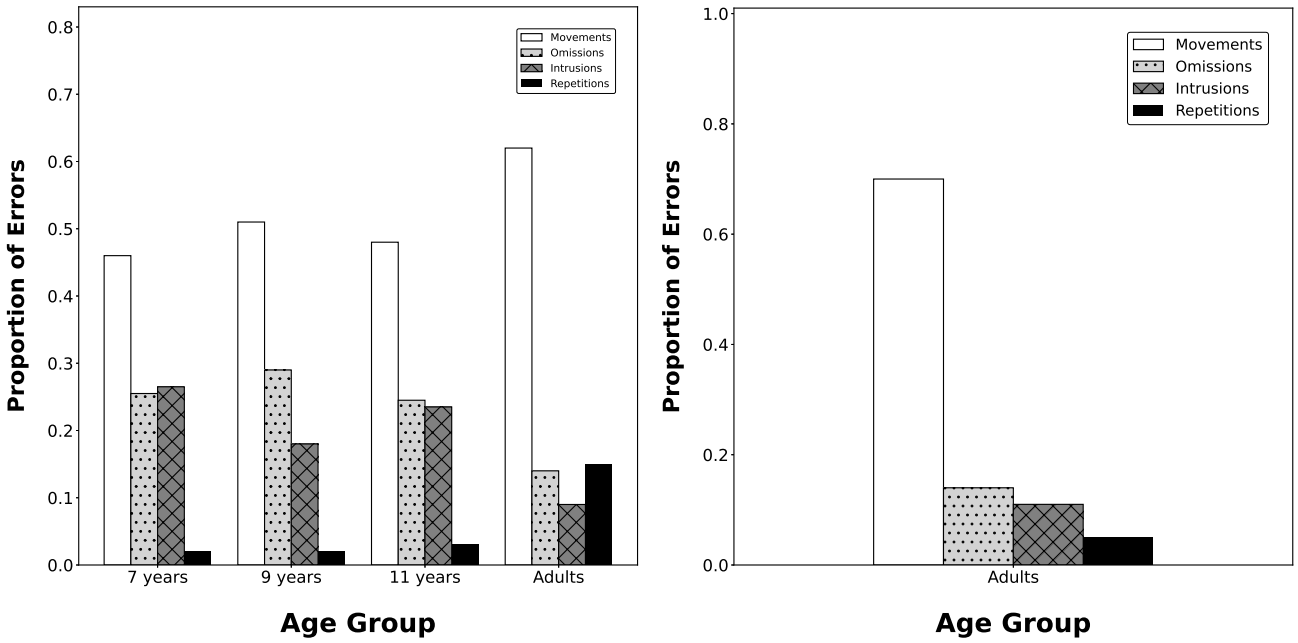
It is also not wise to use a very high threshold as the model would have to be extremely confident in its prediction to recall a letter. This would lead to a model where almost all the errors would be omissions.

According to the first point, a good threshold value is around 50%. This value also makes sense, as it would correspond to a situation where the model can make its predictions only if it has a better chance of predicting the correct answer than the wrong one. Therefore, the exact threshold value that will be used

by the model is equal to 51%, as it respects both conditions defined above.

With the addition of this omission mechanism, the output feedback should be adapted to suit the presence of omissions. Indeed, if the model makes an omission, then no prediction should be sent as output feedback to compute the next hidden state, as the model was not able to predict any letter.

The same procedure has been used to compute the different types of errors made by the model. The different proportions of errors are represented in Figure 6.6



(a) Experimental data

(b) Model predictions

Figure 6.6: Proportion of the different types of error, for model trained with the omission mechanism.

The data predicted with the modified model show the presence of omission errors, indicating that a very simple threshold mechanism is sufficient to obtain this type of error. In addition, the proportion computed for this type of error is very similar to human data. This is because the threshold value has been set such that the predictions fit the experiments.

The proportion of the other types of error has also been modified with the introduction of the omissions. Two phenomena affect these modifications. First, since omissions are now possible, the total number of errors increases, uniformly reducing the proportion of the other errors. Then, as the omission mechanism affects directly the model predictions, some transposition, repetition, or intrusion errors could appear as omission errors instead. This leads to a non-uniform decrease of the proportion of the other errors.

In conclusion, in most of the models of serial recall, omission errors are rarely considered. However, it is important to model them, as they are an integral part of the various errors that the model can produce. This model has taken a step in that direction by ensuring that the model produces these errors, with a proportion similar to that observed in empirical benchmarks.

6.5 Bigram frequency

Regularities of the sequential structure have also impact on the recall performance during the task. This regularity effect will be studied through bigram (*Rice and Robinson, 1975*). As a reminder, a bigram is a pair of consecutive elements. The recall performance will be better for bigram with high frequency. For this model, bigram of the English letters will be studied. The impact of english letters structure in text will be characterized by using two different types of lists: some constrained only by the individual letter frequencies of the language and some reflecting the bigram frequency structure of English.

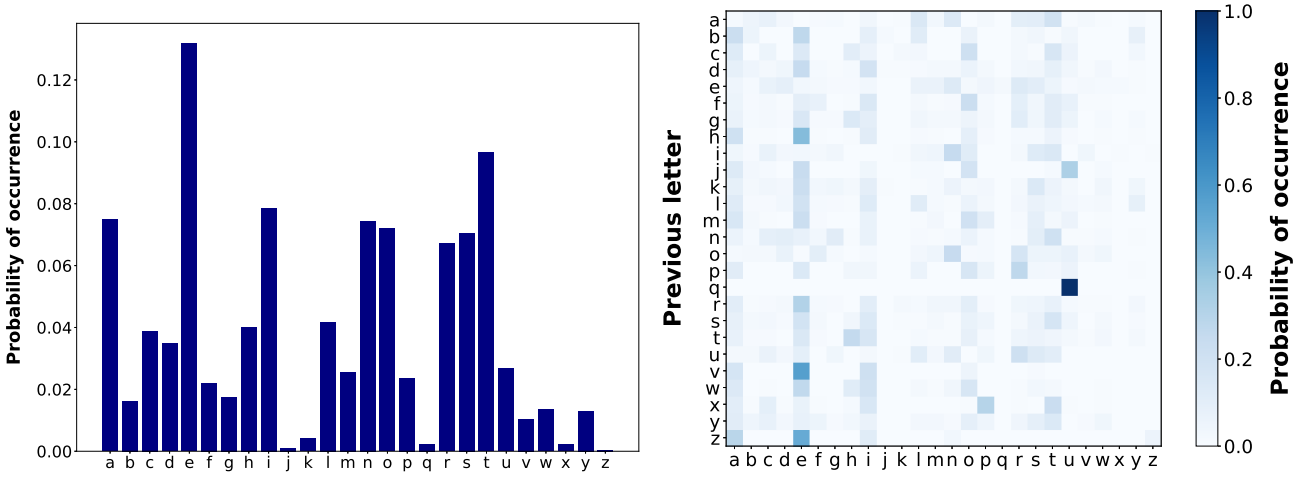
In view of this objective, it is necessary to compute the zero order, i.e. the individual letter frequencies and the first order probabilities, i.e. the letter to letter transition probabilities. To obtain these probabilities, a dataset containing an English structural text has to be chosen. The one that will be used is the Botvinick and Plaut article "Short-Term Memory for Serial Order: A Recurrent Neural Network Model" (*Botvinick and Plaut, 2006*), a small wink to the authors of the first serial recall model using Recurrent Neural Networks.

The article is composed of approximately 13000 words. This remains a subset of all the possible English letters, compared to a lexical containing all the English words. However, this still gives a significant representation of the structure of English words. Moreover, it is not necessary to obtain the precise probabilities of the English letters to reproduce the experiments, but rather values for the different probabilities. These probabilities can be computed using the following expressions, depending if the zero or first order is considered.

$$P(letter\ i) = \frac{Number\ of\ letters\ i}{Total\ number\ of\ letters}$$

$$P(letter\ i|letter\ j) = \frac{Number\ of\ letters\ i\ after\ the\ letter\ j}{Total\ number\ of\ letters\ j}$$

All the different computed probabilities can be grouped together to be better visualized, resulting in a probability histogram and matrix, as represented in Figure 6.7.



(a) Zero order letter probabilities

(b) First order letter probabilities

Figure 6.7: Probability distribution of the letters. Left Figure, the x-axis corresponds to the different letters and y-axis to the probabilities. Right Figure, the x-axis is the different letters, on the y-axis the previous letters and the values are the transition probabilities.

These probabilities show that some letters are more likely to appear in English words, such as the letters E, U, I, A, etc. This same type of behavior can also be observed for first-order probabilities. With the probabilities computed, two different simulations will be considered to analyze the impact of structural data on the serial recall performances.

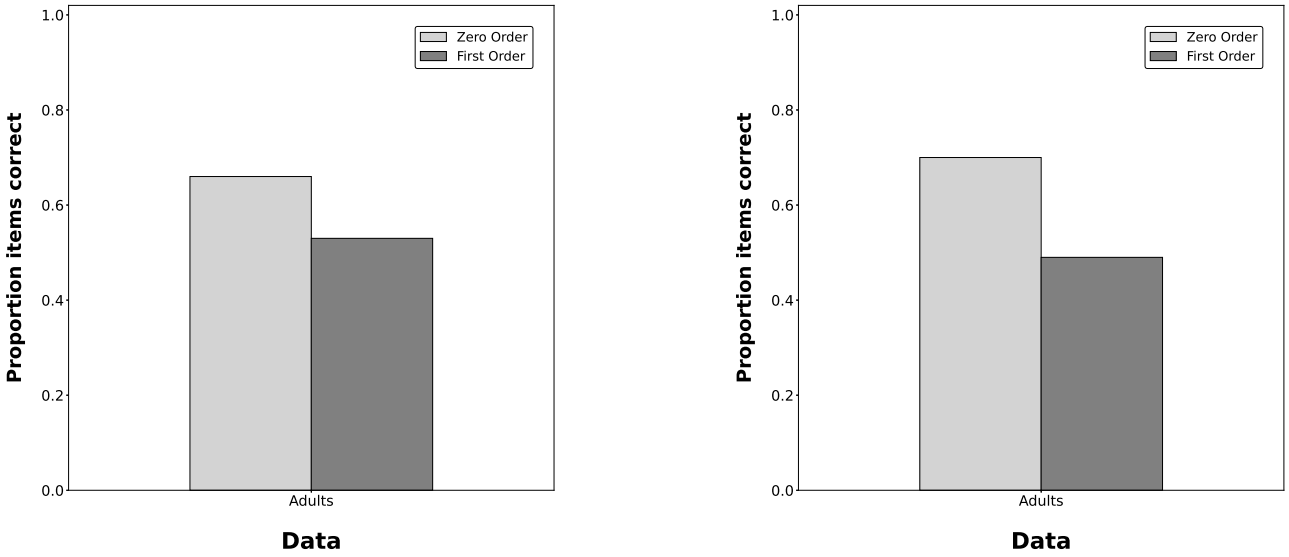
6.5.1 Impact of Bigram

The first simulation will be used to investigate the impact of using bigram on the proportion of lists correctly recalled. Indeed, sequences created using this bigram should have better performances, as they rely on structural regularities, as observed in real experiments.

To reproduce these experimental data, the model was trained on sequences of length varying from one to eight, with the sequences created using the letter-to-letter probabilities. To be in line with the experimental data, the model is trained until the accuracy on lists created using the zero-order or first-order probabilities reached 59% (*Baddeley, 1964*). The model used 75000 cycles to reach this value.

The test set was composed of 5000 sequences created using zero-order probabilities and 5000 sequences created using first-order probabilities. The letters are chosen from the probability distributions. However, repetition of letters in the sequence is still not possible. This means that, if the probability distributions returns a letter that is already present in the sequence then a new letter is drawn. The mean accuracy on both types of sequence is computed during model training.

Once the model trained, it can be used to assess the impact of the bigram on the recall performances. For this, the proportion of correctly recalled lists of eight items created using zero-order probabilities is compared to the proportion of correctly recalled sequences of the same size, but created using first-order probabilities. The results of the model are represented in Figure 6.8.



(a) Experimental data

(b) Model predictions

Figure 6.8: Proportion of item correctly recalled for sequences constructed using zero order probabilities and for sequences constructed using first order probabilities, experimental figure adapted from *Baddeley, 1964*.

This figure shows that the model has better recall performances when the sequences are constructed using a bigram structure. The proportions of items correctly recalled are very similar to the experimental

ones, the model correctly recalls 70% of the sequences created using first-order transition probabilities, whereas only 48% of the zero-order sequences are correctly recalled. The model is therefore able to capture the structure of the sequences and use them to its advantage to make better recall performance.

6.5.2 Impact of Bigram frequency

The second simulation will try to investigate the effect of the Bigram frequency on the recall, as observed by *Kay, 2001*. This experiment used the following small set of consonants: C, D, F, H, L, N, R, S, and T, to create sequences of nine letters long.

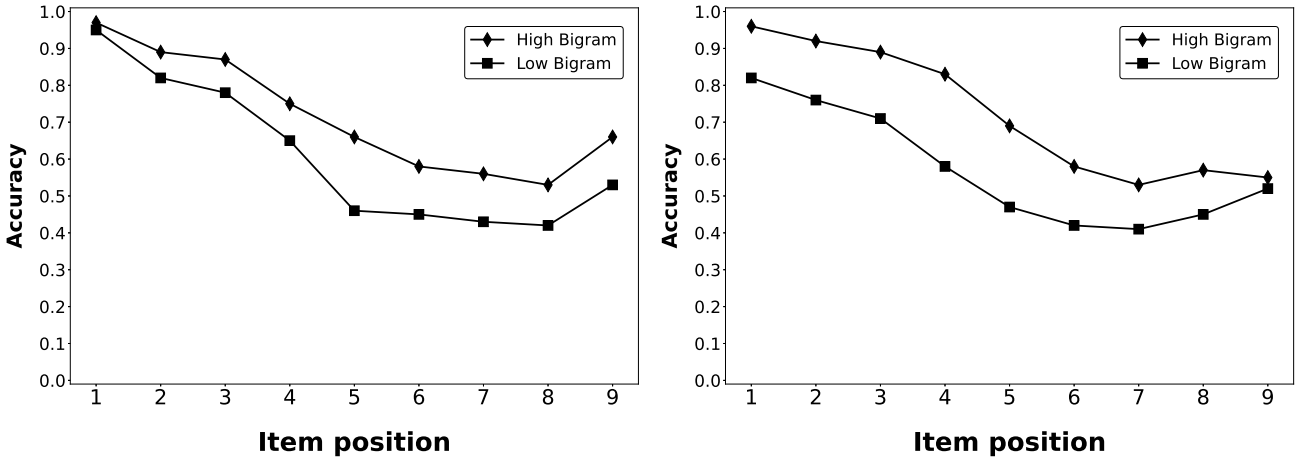
To reproduce this effect, the network has been trained on sequences composed of one to nine items, using only the small subset of consonants as in the experiments. The sequences are still constructed using the first-order transition probabilities of the letters, and the learning of the model was stopped when the positional accuracy, i.e. the proportion of items recalled in the correct position, reached 67%. This occurred after only 6000 cycles.

A really small number of cycles was necessary to train the model, compared to the first simulations. This mainly results from the smaller number of possible letters to create the different sequences, i.e. nine instead of twenty-six. As the number of possible sequences is much smaller than before, it is necessary to check that not too many sequences were used during model training. The total number of sequences of length 9 created from a subset of 9 letters is equal to the following expression.

$$\text{Total number of sequences of length nine} = \frac{9!}{(9-9)!} = 362880$$

This results from the use of 1.65% of sequences of nine items during training, so the model is not subject to over-learning.

To reproduce the effect of the experimental data, the sequences of length nine were separated into different groups based on a median split of the summed bigram frequency, i.e the sum of all the first-order probabilities of the letters in the sequences. This results in two different groups, one for high bigram frequency, the other for low bigram frequency. The results obtained by the model are represented in Figure 6.9.



(a) Experimental data

(b) Model predictions

Figure 6.9: Serial position recall curves for sequences of low and high bigram frequency. The x-axis is the item position and the y-axis the probability of recall.

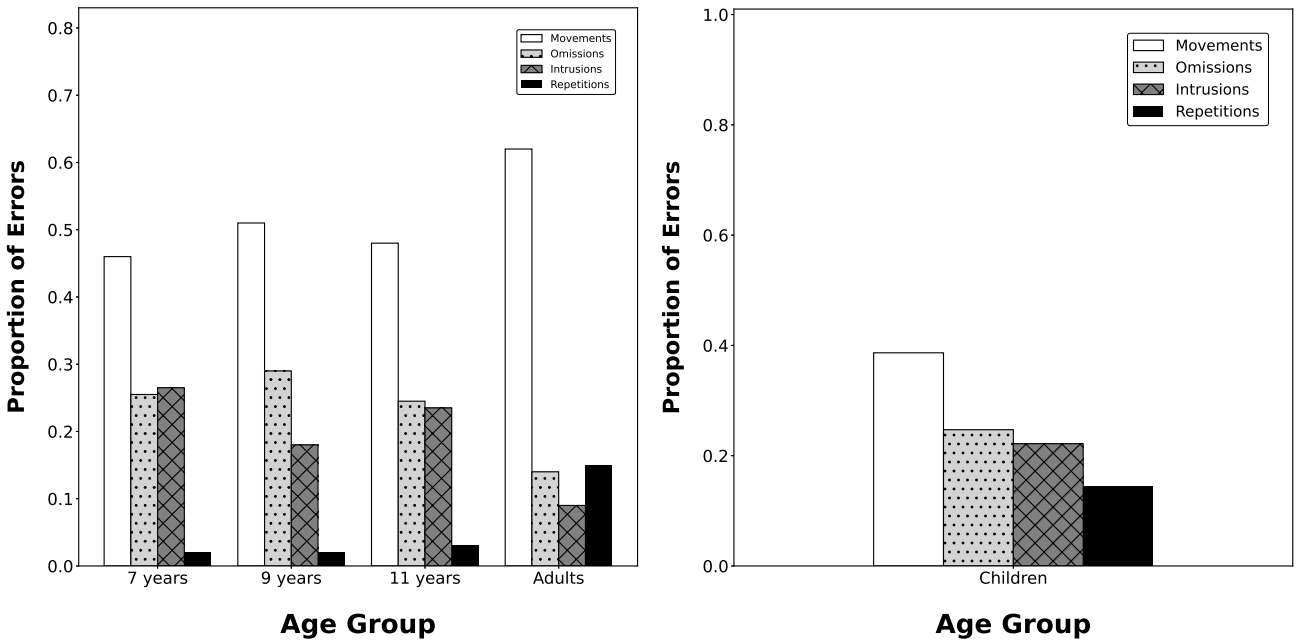
As expected, the sequences with a high sum of bigram frequencies have better recall performance for all the different positions in the sequences. However, there are two main differences with the empirical benchmark. First, there is an important difference between the recall performances of the model on high and low bigram for items in the beginning of the list, whereas this difference should be small at the beginning and increases with the item position to fit human data. The second point is that there is no recency effect for the last item. In addition, the model performance for this item on low and high bigram is almost identical, which should not be the case. In conclusion, the model is able to reflect the principal behavior of bigram, which is that recall performances are better for high bigram, but the serial position recall curves do not match very well with the experimental ones.

6.6 Age effects

Until adulthood, the human brain undergoes changes in its development, which ultimately affect the performance on the serial recall task. In fact, children tend to have worse recall performance than adults on the same task (*Walen, 1970*). These differences in age also have an impact on other effects, such as the proportion of errors and the transposition gradients (*McCormack et al., 2000*).

As the recurrent neural network model also undergoes a learning procedure, it is possible to reproduce the effects of age on the task. For this, we limited the analysis on the distinction between children and adults, but did not distinguish different ages for children.

To represent this effect, the only part of the model to be modified is the stopping criterion. Adults are globally capable of recalling 58% of sequences of six items, and the learning phase of the model is stopped when this condition is met. For children, this value is much lower. However, this value depends on the age of the children. For this reason, we considered a case study in which children can correctly recall 30% of the sequences. The stopping criterion has been set to this value and the model learned using the same training procedure until reaching this value. The proportions of error obtained for this new model are represented in Figure 6.10.



(a) Experimental data

(b) Model predictions

Figure 6.10: Proportion of the different types of error, for children.

The proportion of transposition errors has decreased, as this is the case in human data. However, the number of repetitions is too important when it should be close to zero. The evolution of the proportion of transposition distances with age can also be assessed.

The predictions showed a higher number of transposition errors of a distance of two in children, which is in line with the empirical data. However, for both adults and children, the errors of distance two are much smaller than the ones observed in human data. The model predicts 90% of transposition errors of a distance of one, and almost everything else are errors of distance two, whereas the last percentage should be higher to fit the experiments. In addition, the model produces almost no errors of larger distance. This could be explained by the output feedback, which directly favors the impact of errors on the next timestep.

In conclusion, the effect of age can be partially taken into account using this type of model. One could reproduce this effect more plausibly by determining the proportion of sequences correctly recalled in children to set the stopping criterion of training to this value.

6.7 Confusability

All previous results have been obtained using independent elements. However, in reality, there could be some relations between the letters in the sequence. The one we will be trying to reproduce is the notion of similarity. To take this feature into account, the representation of the elements in the sequence has to be modified.

The task has been modified to take into account the similarity feature between elements. This has been done by separating the input and output layers into two different groups of units, one for the item representation and the other group for the similarity representation.

For the item representation, the sequences can be created using 36 different elements. This means that the elements of the sequences cannot be directly related to letters anymore, but is more abstract in this case. However, this makes absolutely no difference to the task in hand. Each unit is unique to a particular element.

The similarity is represented using six different units. Each unit is shared by six different letters of the set. Therefore, each item overlaps with five other items, all of them are confusable items, and did not overlap with the remaining 30 elements, they are non-confusable items.

Except the representation of the items in the sequence, the task is constructed in the same way as in the general model, using both the recall and end of list items. Sequences of lengths from one to six have been created randomly without any repeat of items in the sequence. This means that the sequences used to train the model contain both confusable and non-confusable items in an arbitrary mixture.

Before being able to train the model using the new item representation, the network has to be adapted so that it can handle both features of the items, item representation and item similarity.

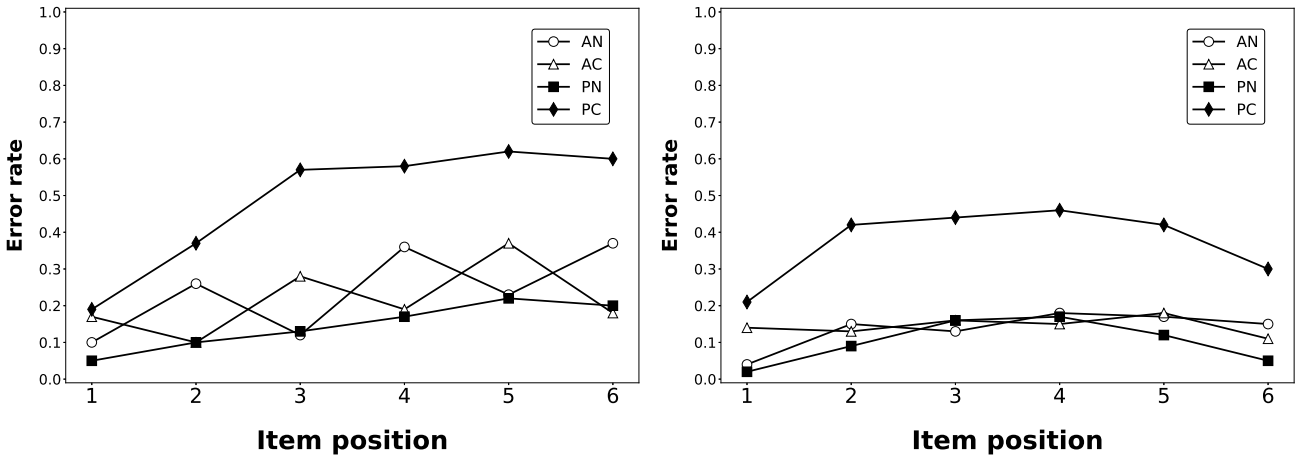
The first modification done in the network concerns the application of the softmax function to the output layer. Indeed, if this function is applied directly to the whole output layer, then the probability distribution obtained would not have any sense as it would result in one single probability distribution for the item and similarity units. Therefore, a softmax function is applied to each group of units, such that the model output the probability distribution of the items and the probability distribution of the similarity.

The mechanism used by the model to predict an item also has to be adapted. If the maximum activation in the output layer is used, then the model could predict a similarity feature instead of an item. If the model predicts the maximum probability in the group of item units, then the model will not take the similarity feature into account. Thus, another mechanism has been introduced. The prediction is chosen by computing the alignment between the output activity layer and the binary representation of the different elements, including both the item and the similarity feature. The alignment between these two vectors is done by computing their dot product. The prediction of the model corresponds to the element with the maximum alignment.

The last modification of the network is in the loss function, which is essential to update the model's weights. Indeed, the KL divergence takes probability distribution as arguments, the whole output layer cannot be used directly. As for the softmax function, the loss is applied separately to the item units and similarity units groups. Then, these two losses are added, such that both contributions are used to update the weights of the model.

After all these modifications, the model is finally able to be trained the task, until the average accuracy on pure confusable and pure non-confusable sequences reach 34%, as in the experiments (*Baddeley, 1968*). Training was completed after 110 000 cycles, much longer than the previous simulations as the introduction of similarity make the task more difficult to learn.

To compute the different experimental curves, the model will be tested on pure confusable, pure non-confusable, alternation confusable non-confusable (AC) and alternation non-confusable confusable (AN) sequences. The difference between the two last types of sequences is whether the sequence starts with a confusable item (AC) or a non-confusable item (AN). The recall curves will be expressed in terms of error rates instead of recall probability, for better visualization of the saw-tooth pattern. These new curves are totally equivalent to the previous serial position recall curves as the error rate is defined as one minus the recall probability. The results are represented in Figure 6.11.



(a) Experimental data (b) Model predictions

Figure 6.11: Serial position recall curves of lists composed of pure confusable items (PC), Pure non-confusable items (PN), alternation confusable non-confusable (AC) and alternation non-confusable confusable (AN). The x-axis is the position of the item and the y-axis is probability of recall.

Some of the serial position recall curves created by the model are very different from the experimental ones. For the sequences of pure non-confusable items, the curve is similar to the experiments. This was expected as this corresponds to the classical forward serial recall task investigated in the previous simula-

tions. For lists of pure confusable items, the error rates are much higher than for non-confusable as the items are more similar. These error rates are smaller than the ones observed in the different benchmarks. In addition, the model predicts a too important recency effect. This serial position recall curve does not match the experiments for the last items of the sequence.

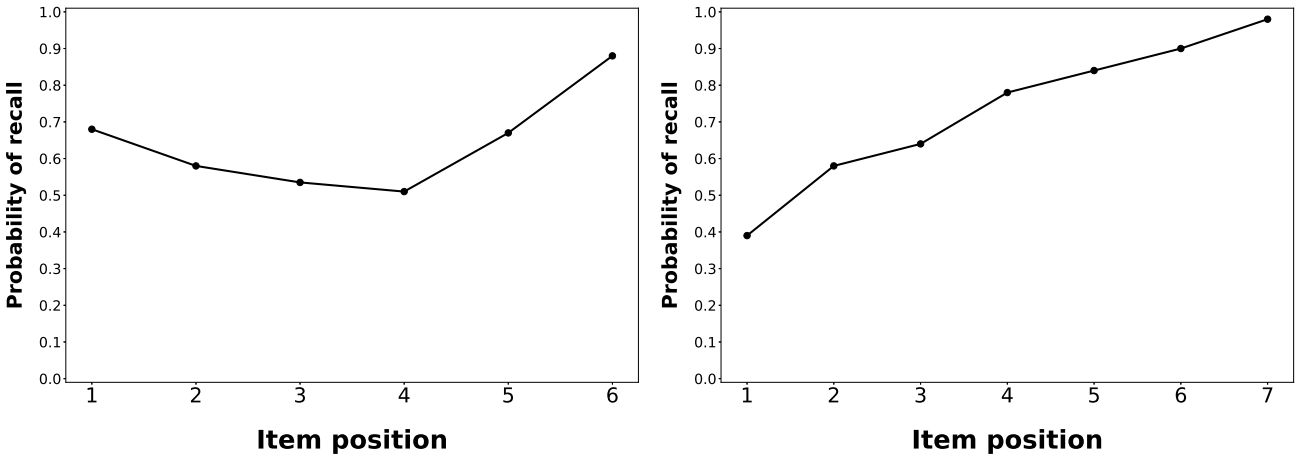
The lists of alternating similarity are the most divergent from the experiments, as they almost do not present the characteristic saw tooth pattern of this type of sequences. The two curves are closer to the pure non-confusable items recall, which means that introducing a confusable item between two non-confusable ones has almost no impact on the model performance. This is not in line with the experiments, as the confusable items should lead to a higher error rate.

6.8 Backward recall

It has been shown that the model can reproduce many experimental results of the serial recall task. However, there also exist different variations of this task. It could be assessed whether the model can also be used to these variations. For this, this section will focus on backward serial recall.

Backward serial recall is exactly the same task as the one used previously, except that items must be recalled in the reverse order in which they were presented. This task presents a high recency effect and a small primacy effect as the first elements encoded are the last to be recalled (*Rosen and Engle, 1997*). The same model and procedure were used to train the network for this task. The dataset has been adapted such that the true outputs correspond to the backward recall sequences. The model was completely trained after 40000 cycles.

The recall curve for this task will be investigated. The curve predicted by the model is represented in Figure 6.12.



(a) Experimental data (b) Model predictions

Figure 6.12: Serial position recall curves for backward recall. The x-axis is the item position during encoding, the y-axis its recall probability. Experimental data adapted from *Kowialiewski and Majerus, 2024*.

The plotted recall curve is such that the first item position is the first item in the presented sequence. It can be seen that the serial position recall curve is completely different from the human benchmarks. The curves has high performance for and then present a monotonic decrease of its performance towards

small recall probabilities for increasing item positions in the sequence.

This means that the model is not able to capture the small primacy effect and to correctly remember the different items of the sequence as the model performance are much smaller than in experimental data. This experiment highlights that the Elman cell is not able to keep in memory items during a long period of time. Indeed, for the most extreme case, the first item presented is kept in memory twice longer than in the forward recall, as it is the last item to be recalled in the backward task.

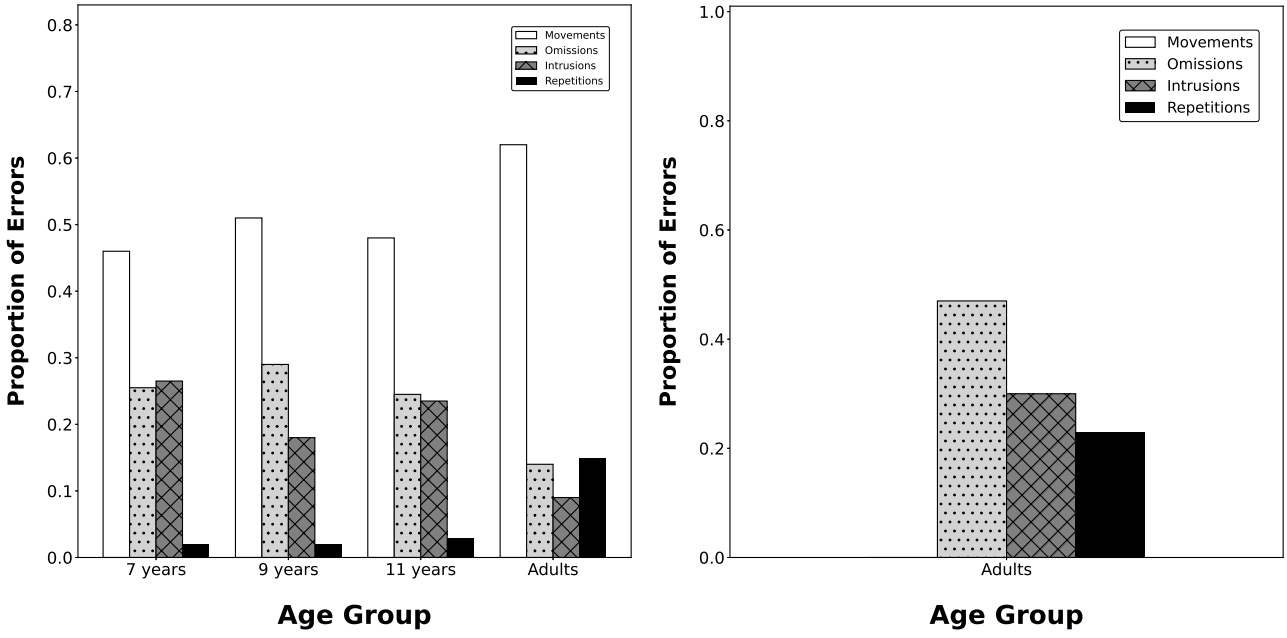
6.9 Importance of output feedback

The output feedback is one of the mechanisms added to the classical version of the recurrent neural networks as a necessary condition to reproduce some experimental results. To characterize the importance of such feedback, the model has been trained without it, and the results of the two versions of the model will be compared. Without the output feedback, the update equation of the hidden states becomes the standard expression used in recurrent neural networks.

$$\mathbf{h}_\mathrm{t} = \sigma(\mathbf{W}_{xh}^T \mathbf{x}_t + \mathbf{W}_{hh}^T \mathbf{h}_{t-1} + \mathbf{b}_h)$$

Except for the removal of this feedback, nothing has changed, either in the structure of the model or in its learning process. All the different results previously analyzed have then been reproduced.

For the sequence-length curve and the recall curve, the results are still very similar to the experimental curves; both curves present their characteristic sigmoidal and bow shape. This implies that the output feedback does not play a role in obtaining these results. This was expected as these curves are characteristics of the global architecture of the network and the model training. Another focus can be put on the proportion of the different types of error made by the model. These results are represented in Figure 6.13.



(a) Experimental data

(b) Model predictions

Figure 6.13: Proportion of the different types of error, for model without output feedback.

A major difference appears between the results of the model without the output feedback and the experimental results, the number of transposition errors made by the model is practically equal to zero. Instead of this type of errors, the model produces many more repetition errors. The proportion of errors is therefore completely different from the experimental results.

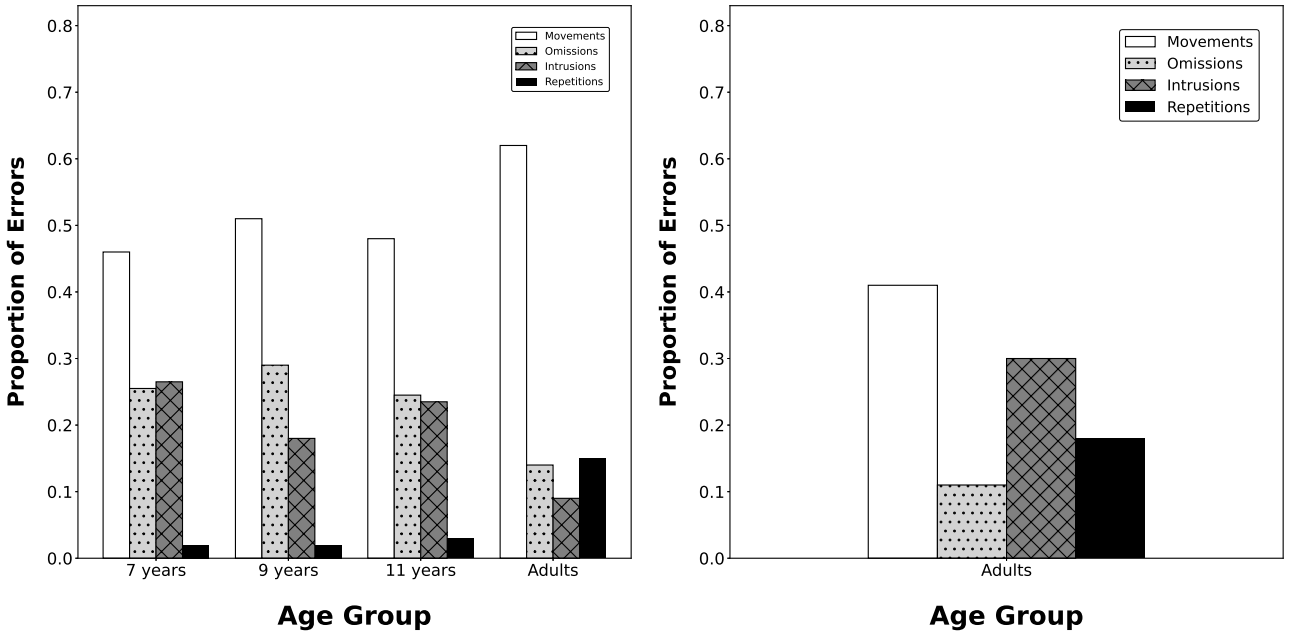
This experiment showed that the output feedback is an essential mechanism to produce the transposition errors. Indeed, when the model made a prediction about the letter to recall, the output feedback allows to use this prediction to compute the new hidden state. If the model recalls a wrong letter, then the predicted letter will be taken into account to predict the next letter, by inhibiting the letter just predicted through this feedback. The model will be less likely to produce two times the same letters, but will recall instead another letter of the sequence, interchanging the position of the letters in the recall sequence and thus producing transposition errors. When there is no output feedback, an active neuron that contributes to the prediction of a letter will remain highly active during several timesteps, such that the model will produce repetition errors.

In conclusion, the output feedback is an essential mechanism for the model to reproduce transposition errors. However, this mechanism does not affect the other experimental curves that the model has to reproduce.

6.10 Effects of teacher forcing

The second mechanism to characterize is teacher forcing. As a reminder, this mechanism corrects the output feedback during the training of the model, i.e. if the model recalls the wrong letter, the correct letter that the model should have recalled is used instead as output feedback.

The use or non-use of this mechanism should lead to different model results. First, the accuracy in the training set should increase more rapidly with teacher forcing than without due to the model correction mechanism. However, since this mechanism is no longer used once the model is trained, the accuracy of the test set should not be affected by this mechanism. In addition, the number of transposition errors should be more important with teacher forcing, as this mechanism makes the model rely more on the output feedback to make its next prediction. The curve of error proportions is represented in Figure 6.14.



(a) Experimental data

(b) Model predictions

Figure 6.14: Proportion of the different types of error, for model trained without teacher forcing.

For these predictions, it can be seen that the removal of teacher forcing allows to have more repetitions and less transposition errors compared to the model using this mechanism. This difference can be explained by the fact that, without teacher forcing, the model is trained with incorrect letter prediction as output feedback, and so will less rely on the output feedback. The intrusion errors are still too important, but the intrusion errors are close to the human data. However, it is not very close to the empirical benchmark.

Finally, apart from the proportions of the different types of error, the other experimental curves should not be affected by the presence or not of teacher forcing, as it has been assessed in the last section that the output feedback only impacts the types of error made by the model. This has been confirmed by the predictions of the model, another evidence that the output feedback only affects the types of error.

Other analyses that will be done in later chapters also showed some evidence against the use of teacher forcing in the model to reproduce the experimental results. For this reason, this mechanism has been removed from the model for the next analyses.

6.11 Conclusion

The model of serial recall using recurrent neural networks can reproduce different empirical effects observed in the real experiments as the serial position recall curves, the transposition gradients, etc. However, there are also effects that the model using a simple cell as the Elman cell struggle to reproduce: confusability, backward recall. The Elman recurrent neural network is a basis to obtain the major effects of serial recall, but should be improved to reproduce many other effects.

Chapter 7

Exploring the cell dynamic

In the last chapter, it has been seen that the serial recall model was not able to correctly reproduce all the results observed experimentally. It is important to assess if this is a limitation of the type of model itself or comes from the dynamic of the neurons in the hidden layer. Until now, the model only used the most simple cell dynamic, Elman cells, in the network. However, more complex cells have been introduced over time. This chapter will study the impact of three new cells: GRU, BRC and nBRC on the model performance. All the defined characteristics and mechanisms of the network have been kept constant, except the update equation of the neurons in the hidden layer.

7.1 Training of the different models

The training procedure of the models remains strictly identical. However, the number of cycles required to train the model to the desired level of accuracy could vary depending of the cells. These values are compared in Figure 7.1.

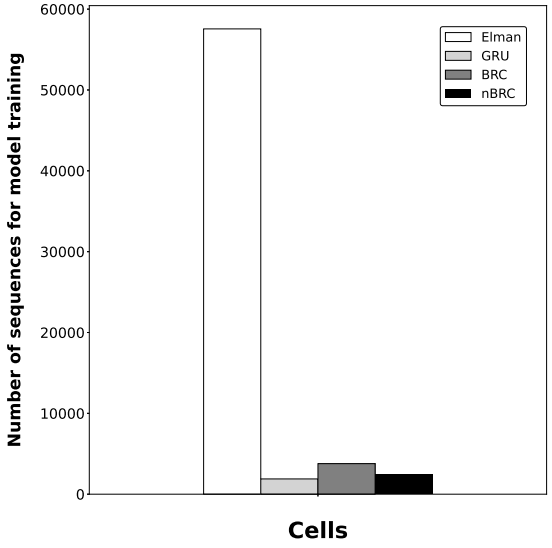


Figure 7.1: Number of sequences needed to train the different cells to the accuracy of 58%.

There is a huge difference between the number of cycles required to train the model with Elman cells compared to the others models. Indeed, all the newly introduced cells are trained after a maximum of 5000 cycles. As a reminder, the total number of sequences composed of six letters is equal to 165 765 600. In this sense, the upper bound for the number of sequences of length 6 used during the training of the new models is given by the following expression.

$$Percentage\ of\ sequence\ of\ length\ 6\ used = \frac{550}{165\ 765\ 600}.100 = 3,3\ 10^{-4}\%$$

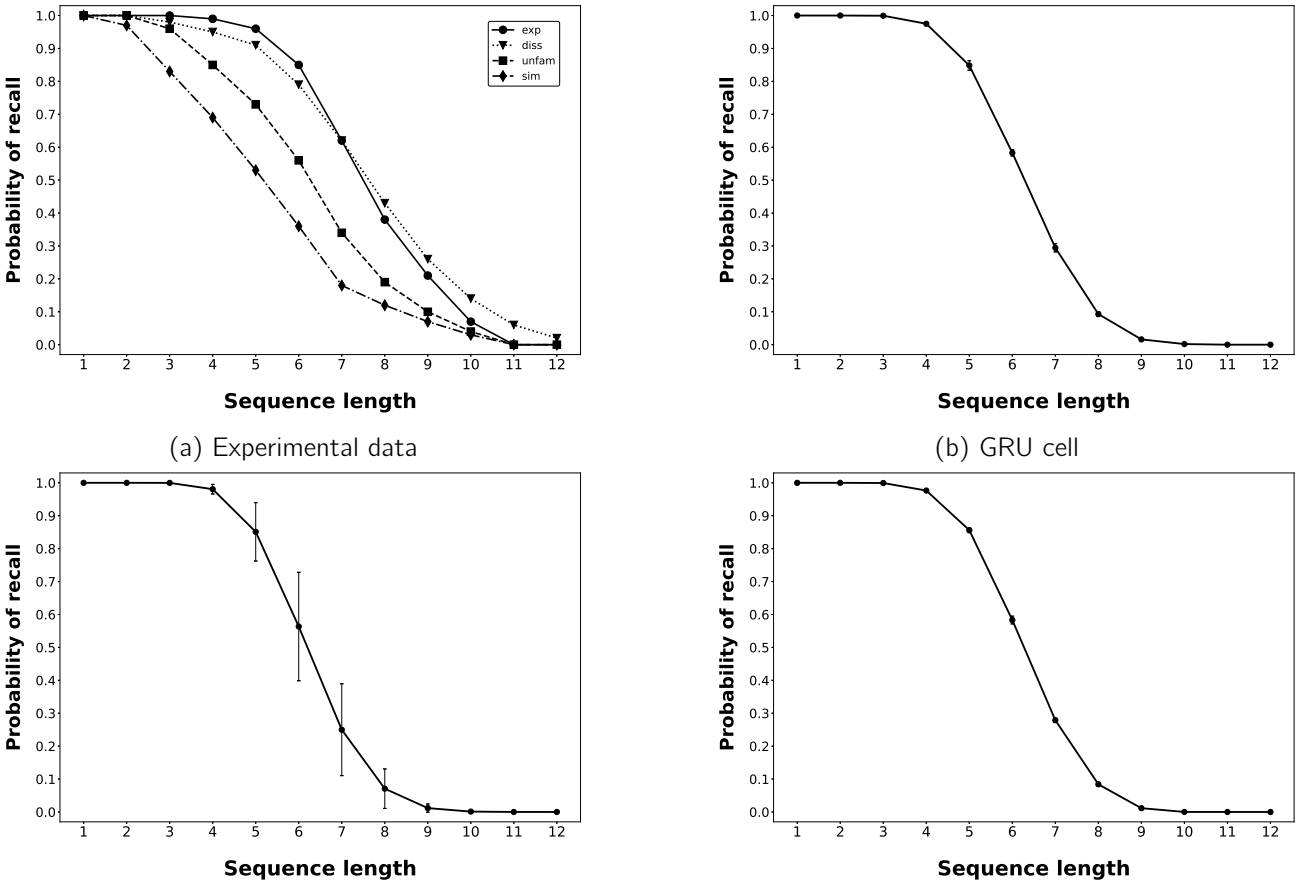
This result clearly shows that the model does not learn sequences by heart how to correctly recall a large set of sequences, which would be the case if a really large number of cycles were used for the training. Instead of this, the small number of cycles used reveals that the model learns how to correctly perform the task, and thus is able to recall correctly sequences never presented during the training. The new cells present a higher plasticity as they learn the task very quickly. This behavior is the result of the new intrinsic property of the cells, which is the gate mechanism. This mechanism, present in the three cells, allows to retain information more easily by being able to choose to keep in memory the relevant information.

7.2 Impact on the experimental effects

After the discussion on the training procedure, the experimental curves produced by the different cell dynamics can be compared to find if results closer to the human results can be produced.

7.2.1 Working memory performance as function of the sequence length

The first reproduced result is the evolution of working memory performance with sequence length. All curves of the different cells are represented in Figure 7.2.



(a) Experimental data

(b) GRU cell

(c) BRC cell

(d) nBRC cell

Figure 7.2: Evolution of working memory performance with length for the different types of cells.

All the models have approximately the same sigmoidal shape as the one observed in experiments, they remember short sequences very well and struggle on longer ones. Even if they have the same shape, they all exhibit variations, more particularly in the transition region, i.e. region of transition between high and low performances. Thus, it is interesting to assess which curve is closest to the experimental one.

First, all models have almost the same recall probability for sequences of six items. This was expected as the stopping criterion for the model learning phase is that the recall probability for sequences of length six is equal to 58%, and so is identical for all models. Very small variations are still however present, which comes from the fact that it is impossible to exactly stop the training at this value.

Small variations are present for sequence length just below the memory span, i.e. sequence length of six. Elman and BRC have a higher probability of recall for lengths of four and five, compared to the nBRC and GRU. Even if this difference remains small, Elman and BRC are closer to the experiments.

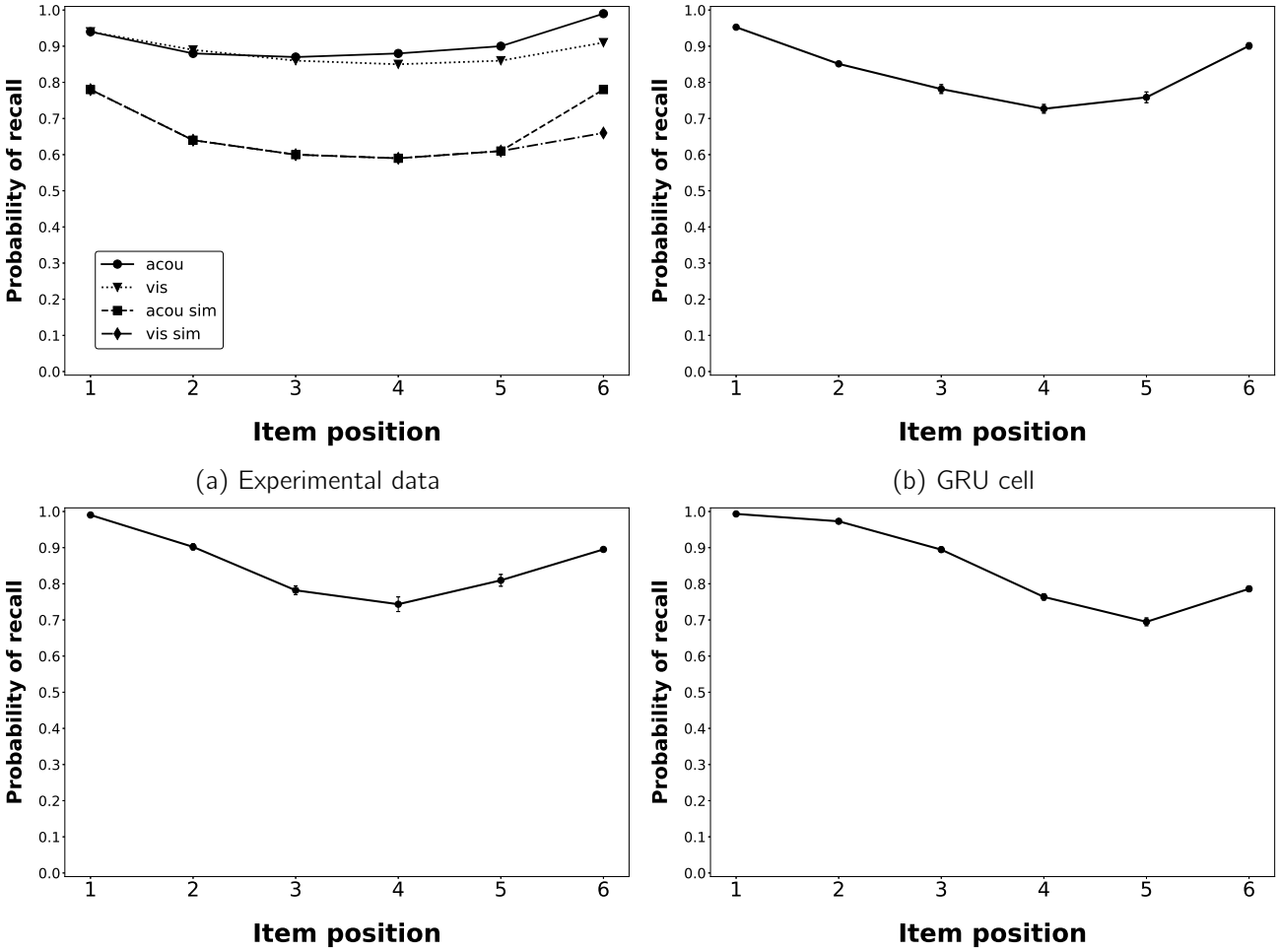
For sequence lengths larger than the memory span, GRU, BRC, and nBRC have a less sharp decrease compared to Elman. These models have slightly better performances for longer sequences, which is more in line with the experimental data.

For very long sequences, all the models have probability of recall almost equal to zero. In Elman cells, the model's recall probability is equal to zero for sequence lengths on which it was not trained. By comparing this probability on sequence length of ten, for the other cells. The maximum recall probability is obtained by the BRC cell and is equal to 0.2%. This means that the network is able to generalize on sequence lengths not used during training, but only have very low recall performance due to the difficulty to remember and recall very long sequences correctly.

In conclusion, all the cells give the characteristic shape of this curve, but the BRC cell has the results that are the closest to the experimental ones.

7.2.2 Serial position recall curves

The serial position recall curve is the most characteristic curve of serial recall, it is therefore essential to check that each cell is able to reproduce this result. The different results are represented in Figure 7.3.



(a) Experimental data

(b) GRU cell

(c) BRC cell

(d) nBRC cell

Figure 7.3: Serial position recall curve of the different types of cells.

All the recall curves present the primacy and recency effects, i.e. the model better recalls the beginning and end of the sequence. Elman, GRU and the BRC cells all have a symmetrical shape as in the experiments. However, this is not the case for the nBRC cell. Indeed, the recall curve for this model is quite asymmetrical. The model has high recall performances for the first elements of the sequences and then an important decrease of them for latter elements. Finally, a recency effect occurs for the last element only. The recall curve of the nBRC model is therefore quite different from the experimental one.

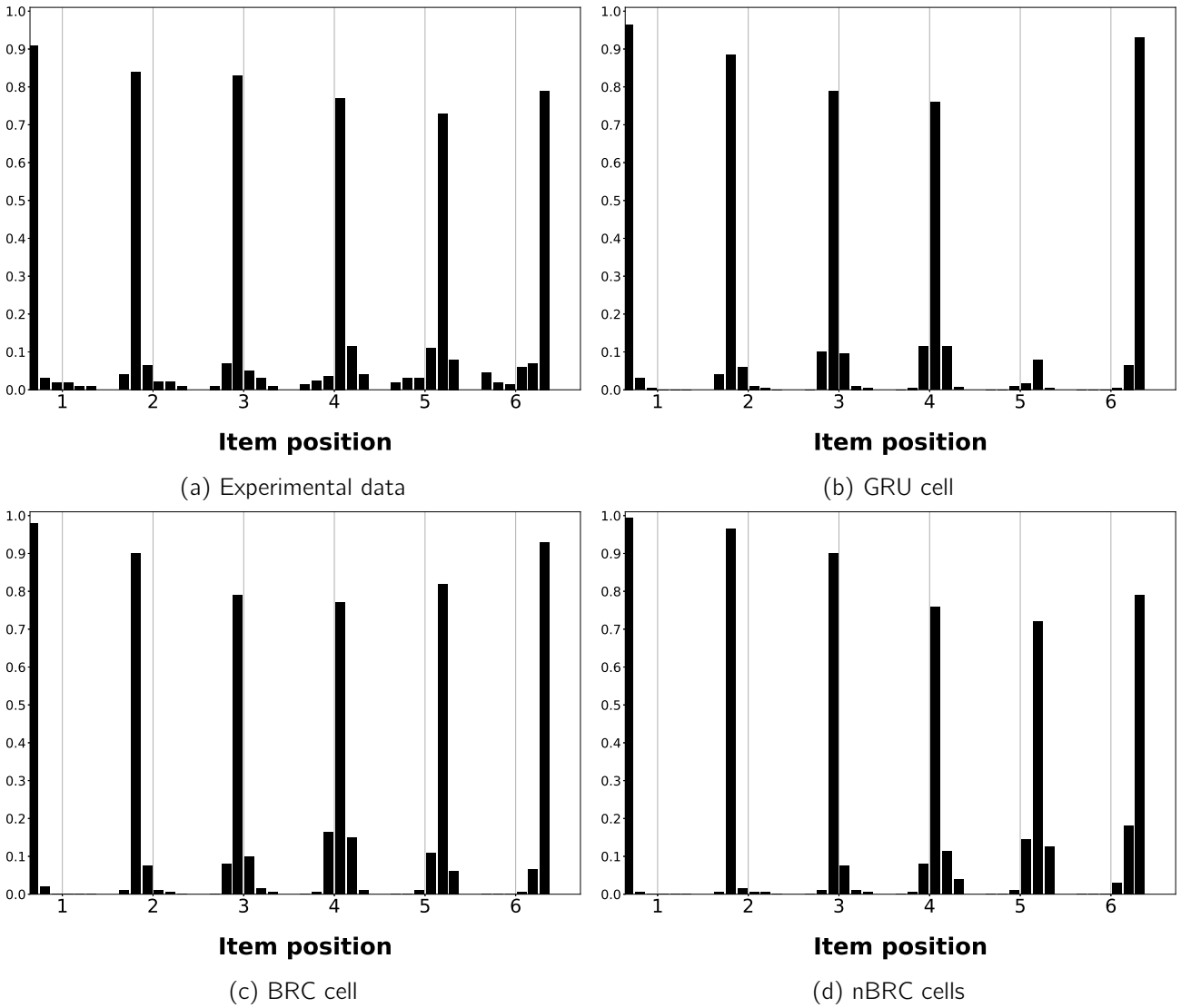
It can be seen that the probability of recall of the first element is almost equal to 100% for BRC and nBRC. This means that these models almost always correctly recall the good letter, which is a bit higher than in the experimental results. There are also variations in the recall probability of the elements in the middle of the sequences. However, the values are still close to the experimental recall probabilities.

Finally, the GRU and BRC present a higher recency effect, close to what is observed experimentally. Whereas, in the Elman cell, the decrease of recall probability with the item position is less important, but presents a smaller recency effect.

In conclusion, except the nBRC cell, which exhibits a too strong asymmetrical shape, all the other cells have recall curves close enough to the experiments, each with its own variations.

7.2.3 Transposition gradients

The last essential effect that will be compared with all the cells is the transposition gradients. Next, the results will be discussed and it will be determined which cells will be analyzed in greater detail to represent the other experimental effects. The transposition gradients of the different cells are represented in Figure 7.4.



(a) Experimental data

(b) GRU cell

(c) BRC cell

(d) nBRC cells

Figure 7.4: Transposition gradients of the different cells.

From all these transposition curves, it can be seen that the nBRC model is the furthest from the experimental data. Indeed, there are almost no transposition errors for the first two items of the sequence. The other models have results quite similar to the experiments. It can still be noticed that the BRC has smaller transposition for the first item. Finally, all the models make much smaller transposition errors of distance equal to or greater than two. This is because the output feedback strongly influences the direct next prediction.

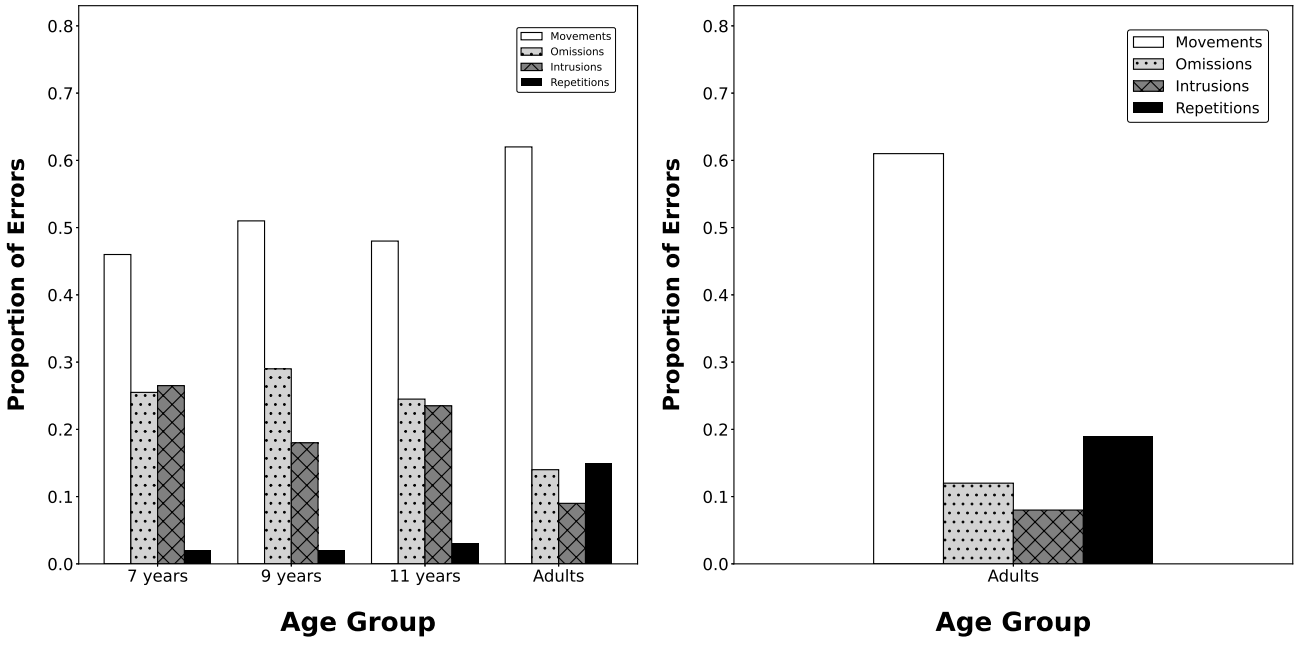
In conclusion, the comparison of the three main effects of serial recall has enabled to highlight differences between the different models. The nBRC gave by far the worst experimental results. For this reason, this cell will not be further investigated to reproduce the rest of the results. This will also be the case of the GRU cell, for a totally different reason. The goal of this chapter is to assess whether biologically more plausible neurons in the network leads to results closer to the experiments. As GRU is a type of cell that has been introduced only for computational reasons, the focus will not be made on this type of cell.

This means that two different cells will be compared, the Elman cell, which results have been described in the last chapter and the BRC, which introduces bistability property and cellular memory.

7.2.4 Types of error

One major investigation of the last chapter concerned the types of error produced by the model. While it has been shown that the output feedback was essential to produce transposition errors and that teacher forcing should not be used to get more accurate results, the proportions obtained by the model still differ from the experiments.

The same experiment has been done with the BRC network, where the omission threshold has been kept to 51%, as for the Elman network, to get a better comparison of the predictions of both models. The results are represented in Figure 7.5.



(a) Experimental data

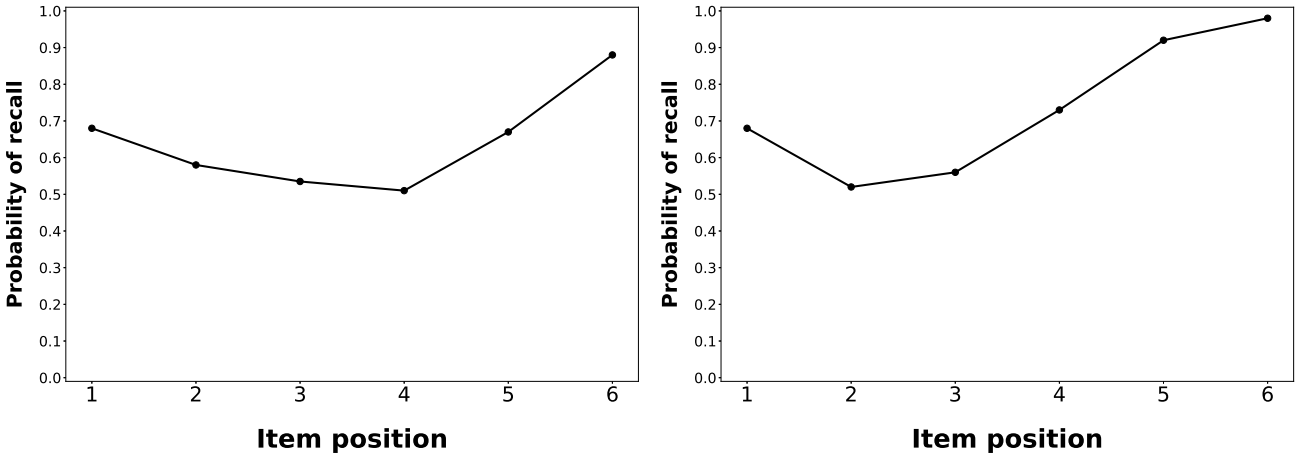
(b) Model predictions

Figure 7.5: Proportion of the different types of error, for model trained without teacher forcing.

The new results almost perfectly match the experimental ones, the global proportion of the different errors are the same and their values are very close. This means that the BRC network better reproduces the experimental than the Elman cells.

7.2.5 Backward recall

The backward recall curve is the result that the Elman network is least able to reproduce. Indeed, the model accurately recalls the first elements of the sequence, but then the decrease of performance as a function of the item position in the sequence does not fit the experimental results. The performance of the new BRC model on this task can be characterized to assess if the new model is able to reproduce such task. The backward recall curve of the BRC network is represented in Figure 7.6



(a) Experimental data

(b) Model predictions

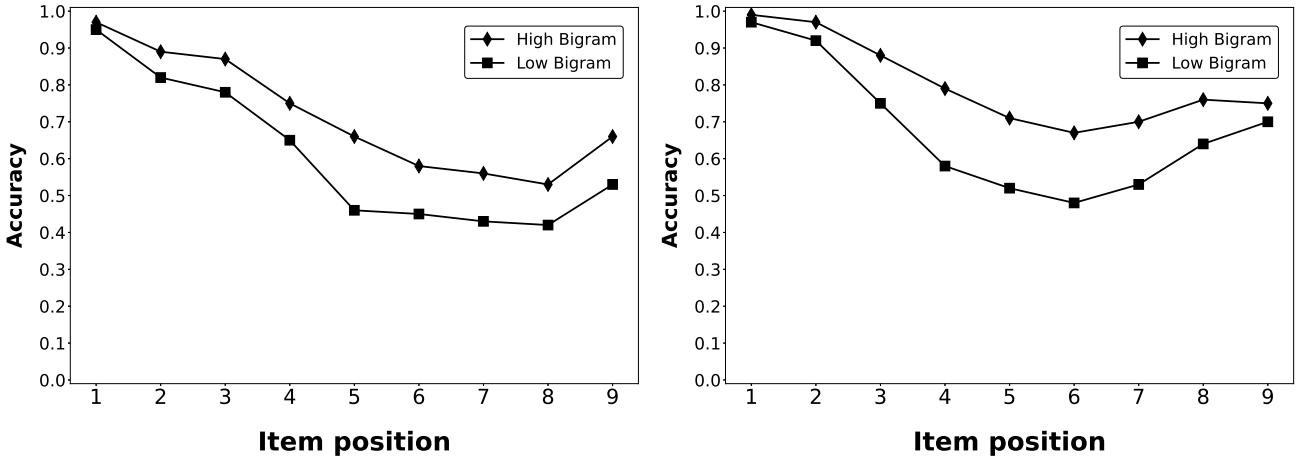
Figure 7.6: Serial position recall curves for backward recall for a BRC model. The x-axis is the item position during encoding, the y-axis its recall probability. Experimental data adapted from *Kowialiewski and Majerus, 2024*.

This new recall curve clearly shows a better curve compared to the one produced using Elman. Contrary to the Elman network, the BRC network is able to learn the task correctly and reproduce its results. The recency effect is highly present for the first recalled items, as they were the last to be presented to the participants during the learning of the sequence. A primacy effect is also observed in the recall curve. This effect is less important than in the forward recall as the first items during the presentation of the sequences are the last to be recalled. Therefore, they have more time to be forgotten.

In conclusion, BRC better performs than Elman on the backward recall task, but the results are still not very close to the experiments.

7.2.6 Bigram

For the bigram effect, two simulations were investigated. For the characterization of the proportion of items correctly recalled, for sequences constructed using zero- and first-order letter probabilities, the Elman cells gave good predictions. Introducing the BRC cell has further improved the predictions, obtaining a proportion almost exactly equal to the empirical references. For the impact of the low and high bigram structure on the serial position recall curve, the results are represented in Figure 7.7.



(a) Experimental data (b) Model predictions

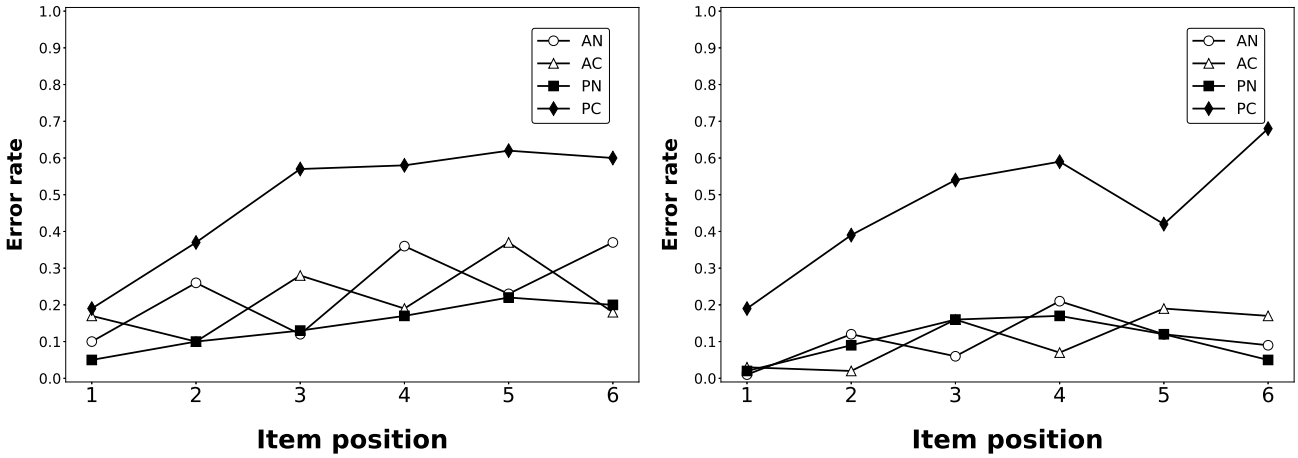
Figure 7.7: Serial position recall curves for sequences of low and high bigram frequency. The x-axis is the item position and the y-axis the probability of recall.

The beginning of the serial position recall curve is better as they are close to 1. In addition, the global trend of the predictions is more accurate, the increasing of recall performance between high and low bigral is more important at the end of the sequence. However, the BRC does not exhibit an increase in performance for the last position of high bigram, which should have been the case due to the recency effect.

Moreover, as this was the case for the Elman cell, the training of the model has important impacts on the recall curves. Caution is therefore called for when analyzing and interpreting these curves.

7.2.7 Confusability

The introduction of confusability is the second effect that the Elman cell model had the most difficulty in reproducing. It is essential to assess whether more complex cell dynamics, such as those found in the BRC, deliver better results. The serial recall curves of the different lists are represented in Figure 7.8.



(a) Experimental data (b) Model predictions

Figure 7.8: Serial position recall curves of lists composed of pure confusable items (PC), Pure non-confusable items (PN), alternation confusable non-confusable (AC) and alternation non-confusable confusable (AN). The x-axis is the position of the item and the y-axis is probability of recall.

For sequences of only confusable items, the error rates are much more important. This is in line with the experiments as the items are more similar. However, there is a decrease of the error rate for the penultimate item, which should not be present. This effect appears through the different simulations launched. No explanation has been found to explain this result, which should not be present.

For the lists of alternating elements, the characteristic sawtooth pattern is much more present. However, the error rates are smaller than the ones obtained for sequences of pure non-confusable items.

7.3 Analyses of network trajectories

The previous section highlighted the differences between the cells in terms of the predictions made by the models. However, this did not give any insight into how the information is processed inside the network. To better understand the dynamic of the models, the trajectories of network can be studied. This corresponds to analyzing how the activation of each neuron in the hidden layer evolves over the timesteps.

However, as many neurons are present in the hidden layer, the analysis of such activity could be difficult as this would result in high-dimensional data. Indeed, 200 neurons has been used in the hidden layer and to execute the task on sequence of length six, 13 timesteps are needed: 6 for the learning phase of the task, 6 for the recall phase and 1 to indicate that the sequence is finished. Combining these information together would result in a data matrix of dimensions 13 x 200. It is not easy to interpret directly such a large amount of data. For this, dimensionality reduction technique has been introduced. An important point to notice is that the network must be trained on the task before being able to do such kind of analyses.

Principal Component Analysis

Principal Component Analysis is a technique used to transform high-dimensional data into a lower-dimensional space. In this technique in particular, the directions in the space, called the principal components, are computed to maximize the variance explained in the new subspace. Once the principal components computed, the data can be projected into them to better visualize its behavior.

7.3.1 Input pulse

The first analysis consists of presenting only one input during several timesteps to the model. After this, the model is left to its own devices, without any new input. This will allow to characterize the behavior of the model when no new information is presented to the network. For this, a letter is randomly chosen and activated during 5 timesteps. After this, no input is given during 200 timesteps. The activation of the neurons are collected and PCA are applied on them. The results of the projection of the trajectories are represented in figure 7.9.

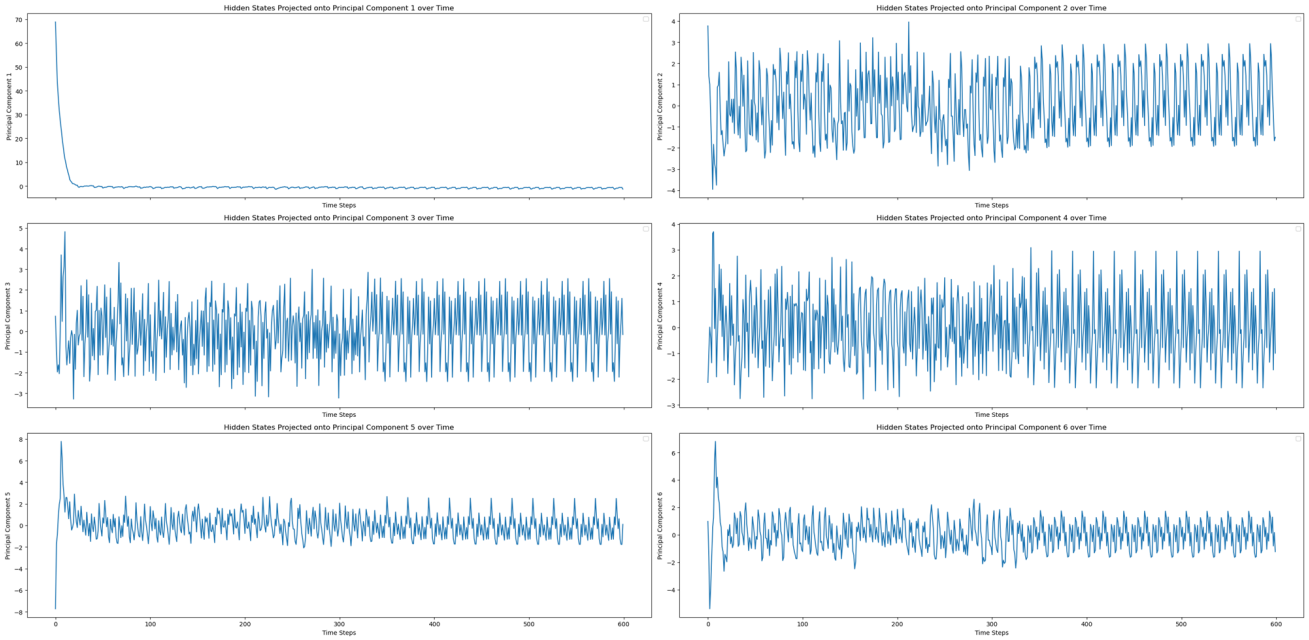


Figure 7.9: Projections of the trajectories of the network when only one input is fed to the network .

This trajectories were computed using a BRC network. It can been seen that the projection on the first component monotonically decrease until zero. For the other projections, the trajectories present large aperiodic variations before stabilizing to an oscillation pattern, suggesting that the network has reached a limit cycle. This is confirmed by the predictions of the model, once in the oscillation phase, the model always predicts the three same items periodically, explaining the oscillation patterns of the trajectories.

This oscillation behavior observed, even when no input is used to fed the network, can be explained by the fact that at each timestep, the network has to predict an item due to its architecture. This item is then used as output feedback to compute the new activity of the neurons in the network. Thus, the model feeds itself even when there is no inputs.

This is not what happens in the human brain. Indeed, humans does not continue to make predictions when the task is finished and no new information are given. The model should be adapted in consequence to take into account this result.

To solve this problem, two different approaches have been considered. The first approach is to let the omission threshold mechanism do the work. If no input is fed to the network, then the model should be uncertain about the items it should predict. Therefore, it should predict an omission and the output feedback would be equal to zero, maintaining a state where the model does not make predictions anymore. The same projections of the network trajectory are represented in Figure 7.10.

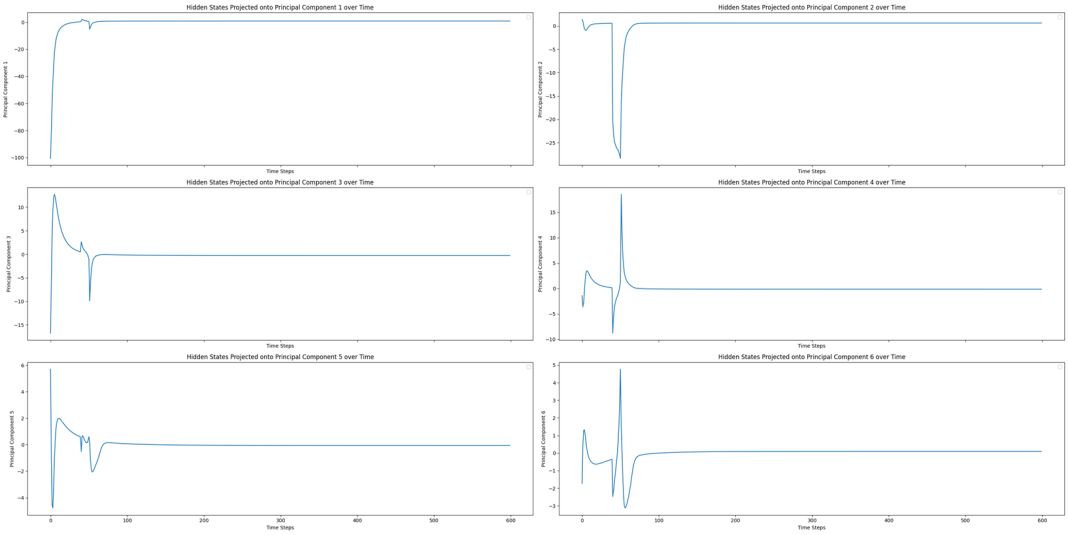


Figure 7.10: Proportion of the transposition distance for adults and children, adapted from *McCormack et al., 2000*.

A first observation is that the trajectories all stabilize towards zero. Thus, the omission mechanism seems to be sufficient to remove the predictions and the output feedback of the model when there is no input fed to the network. However, by looking more carefully at the model predictions, it can be observed that the network still sometimes predicts some items, which therefore will be used as an output feedback.

A solution to this problem would be to decrease the omission threshold. But this is not possible, as it would affect the proportion of the different types of error, which would become different from the ones measured in experiments. The second solution, and the one that will be used, is to artificially remove the output feedback when no input is fed to the network. The trajectories are represented in Figure 7.11.

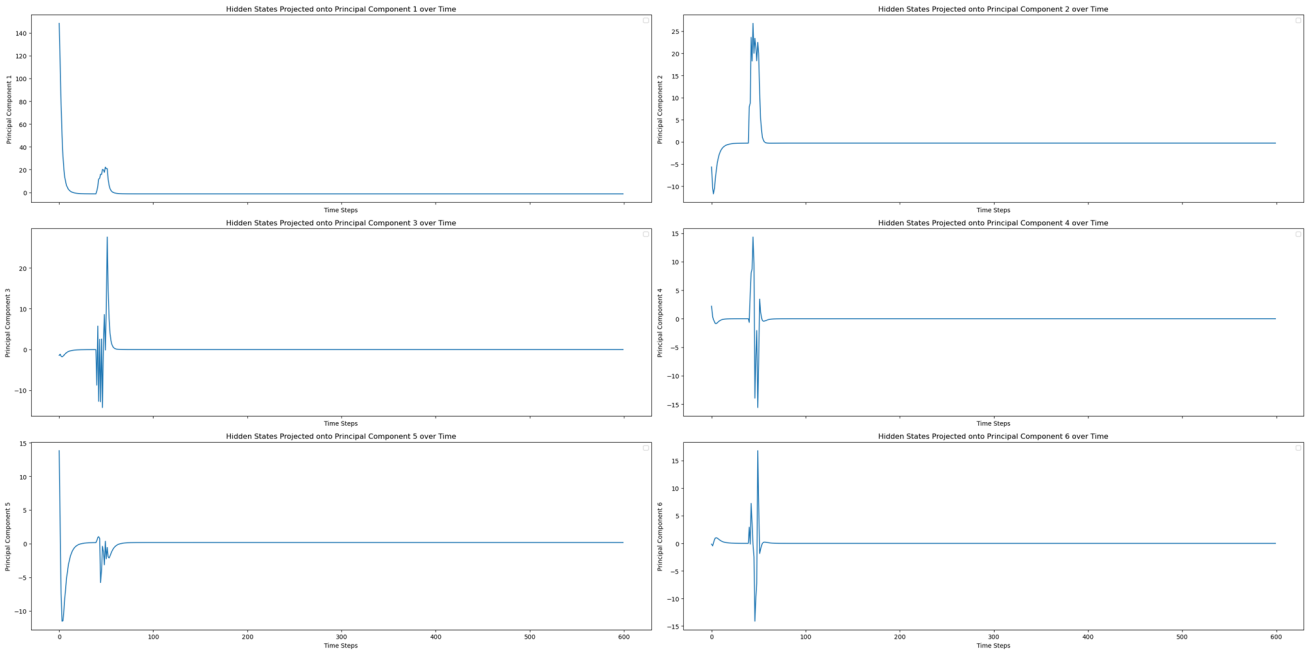


Figure 7.11: Trajectory of the network activity for a BRC network when one input pulse is fed to network and the output feedback is removed when there is no input.

The trajectories clearly show that the model does not predict anything when no inputs are present.

Moreover, the output feedback always remains deactivated. It can be seen that the input applied has only a transient impact on the projection of the network trajectories, i.e. the trajectories diverge from their rest state and then rapidly come back to this state after the input. This seems to indicate that the BRC does not use its bistability property to maintain the last input. The use of longer or larger inputs had no impact on trajectory behavior. The same analyze can be made on the nBRC cell, the results are represented in Figure 7.12.

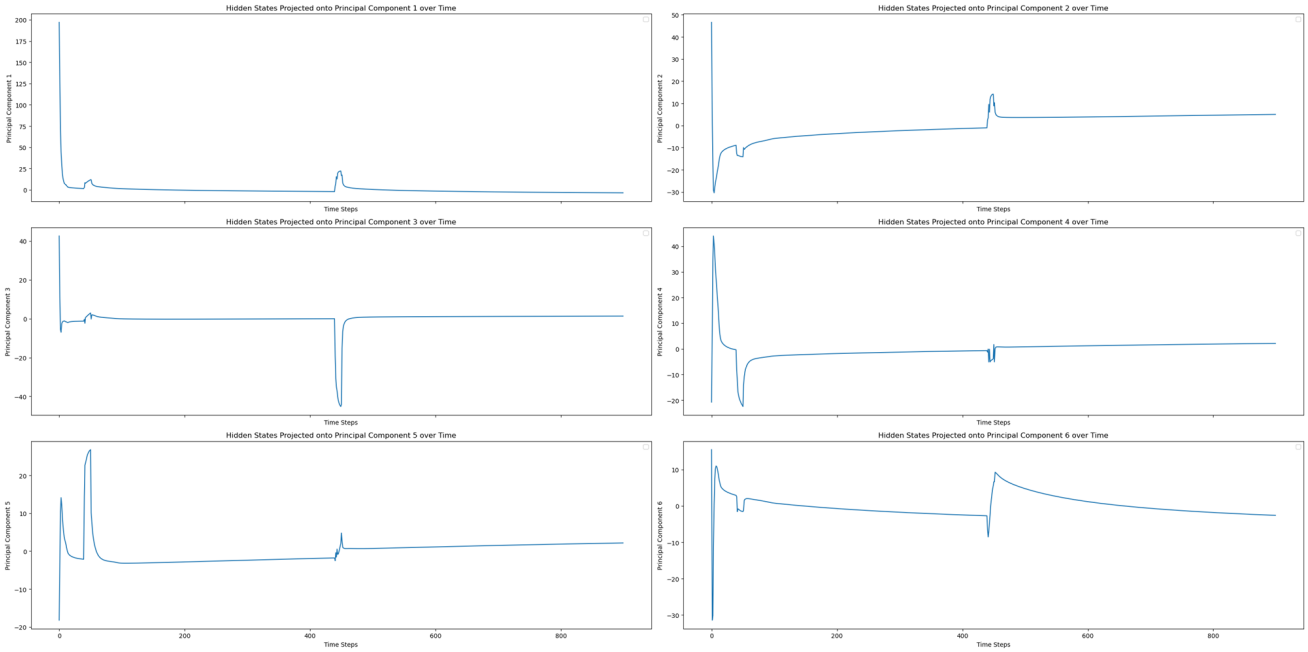


Figure 7.12: Trajectory of the network activity for a BRC network when one input pulse is fed to network and the output feedback is removed when it receives no input, for nBRC cell.

Some projection of the trajectory present a different behavior compared to the BRC. When an input is applied, the trajectories present an offset, that stay over time. With the application of a new input, after a large number of timesteps, the trajectory presents a new offset. The trajectory seems to be able to switch between different states, indicating that the nBRC model could use bistability.

7.3.2 Analyzing trajectories of the sequences

The trajectories of the network during the processing of real sequences can be analyzed. For this, we analyzed twenty sequences of length six. To apply Principal Component Analysis on the data, all the data must be contained into a single two-dimensional matrix.

For this, the computed activity of all the sequences have been grouped one after the other. As the number of neurons of the model is equal to 200, each sequence of length six is composed of 13 timesteps to perform the all task and there are 20 sequences, the total data matrix will have dimensions of 200 by 260. The PCA has been then applied to the neuron activities, and trajectories projected on the components.

If the different sequences are randomly constructed, they are very different from one to the other. This means that the different sequences will be processed very differently by the network, creating different activation patterns of the neurons in the hidden layer. Therefore, this would results in trajectories that are very be different one from each other, and so are the projections on the PCA. It is therefore very difficult to interpret trajectories of sequences constructed randomly, this is why this situation will not be considered.

Instead, to better distinguish and analyze difference in trajectories, sequences that are more similar

should be considered. For this, identical sequences will be considered with only the fourth item of the sequence that vary. This allows to incorporate only one difference across the sequences. Then, the same procedure has been applied and the sequence trajectories are represented in Figure 7.13. In addition, all the trajectories corresponding to sequences that were not correctly recalled have been plotted in black.

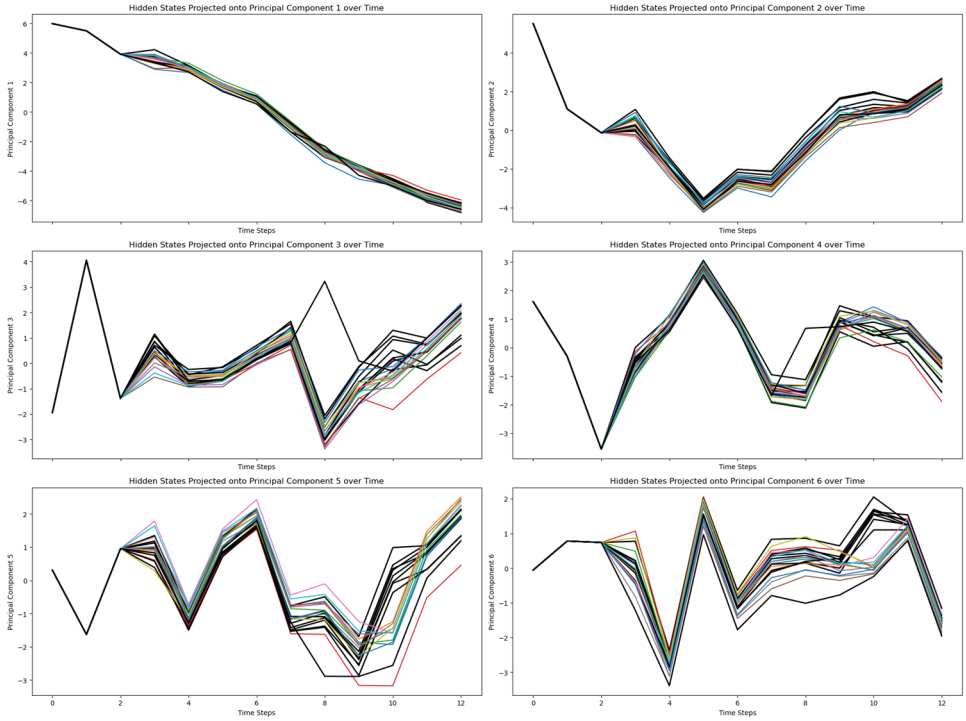


Figure 7.13: Projection of the network trajectories on the Principal Components during the serial recall task. Black trajectories correspond to sequence uncorrectly recalled.

First, it can be seen that the trajectories are identical in a first time, this is because the beginning of the sequences are identical. The trajectories diverge when the different items of the sequences appear. The network separates the sequences such that they can be distinguished, by creating different neuronal patterns and so trajectories. During the rest of the processing of the sequences, the trajectories remain parallel, which is expected, because the sequences are almost identical, their processing should also be the same.

The trajectories corresponding to sequences that are not correctly recalled can be distinguished from those correctly recalled because they have divergent trajectories. The same analyses can be performed on 10 sequences with one varying item and 10 other totally different sequences, with only one varying item at the same position. The results are represented in Figure 7.14.

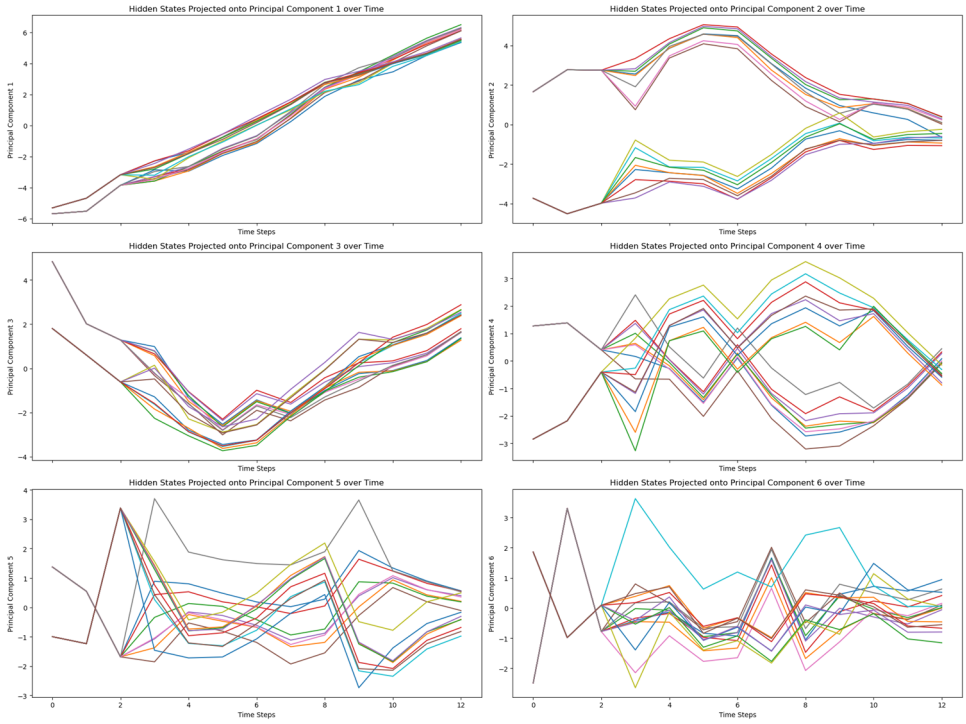


Figure 7.14: Projection of the network trajectories on the Principal Components during the serial recall task, for two completely different families of sequences.

The first component shows that the trajectories of all the sequences are similar, even if the sequences are very different. The other components allows to distinguish the sequences that are very different, by using their trajectories. Finally, the distinction of the sequences with one different item is done the same way as in the previous analysis.

All the behaviors described in this section are the same for the different cells used.

7.3.3 Model behavior after processing of the sequences

Finally, we want to investigate the behavior of the model once it has finished to do the recall task. For this, the same procedure has been applied on sequences with only one letter varying at the same position, which are necessary to do the recall task followed by a period of no inputs. This has been applied to the BRC cell, the projected trajectories are in Figure 7.15.

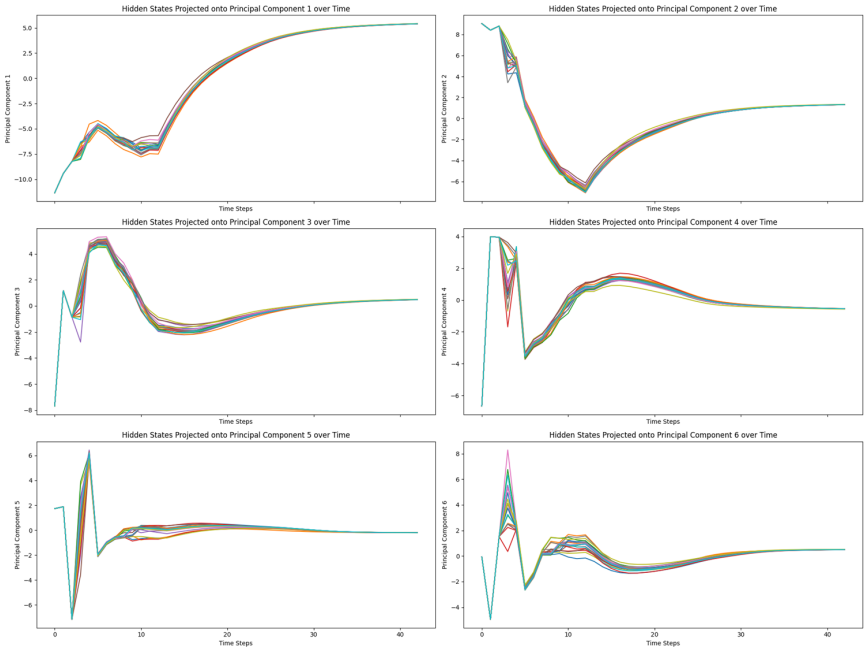


Figure 7.15: Projection of the BRC network trajectories recorded during the task followed by a period of no inputs on Principal Components.

It can be seen that during the task, the network distinguish the sequences by having distinct parallel trajectories as before. However, once the task is done, all the trajectories converge back to the same state. This could indicate that the network is not using its bistability properties. The same analysis can be applied with the nBRC cell and is represented in Figure 7.16.

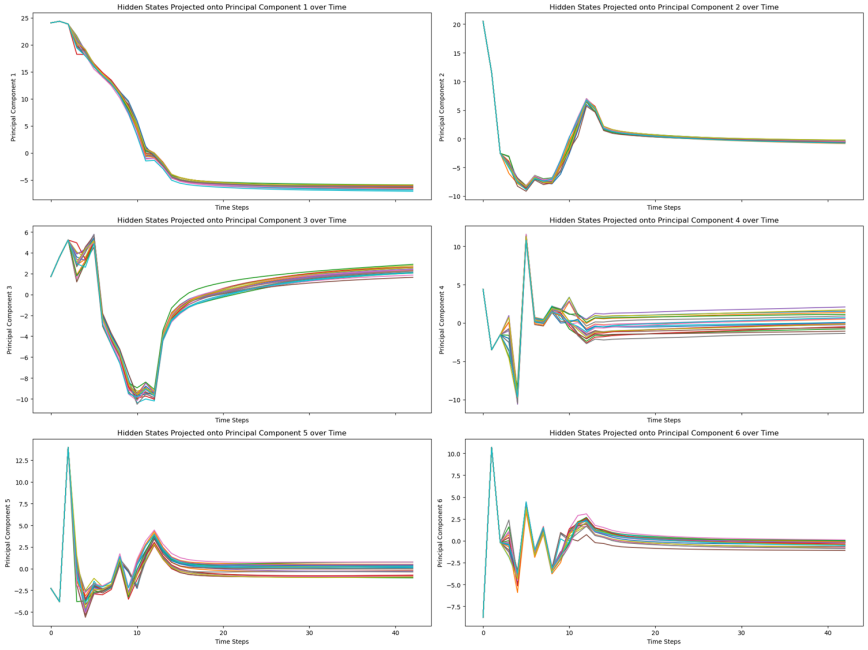


Figure 7.16: Projection of the nBRC network trajectories recorded during the task followed by a period of no inputs on Principal Components.

The same behavior as for BRC cell is observed during the task. However, once the task is completed, the projection of the trajectories remains parallel, distinct and close to the value there were at the end of the task. This could suggest that the nBRC is using bistability to maintain is state when no inputs are

fed to the network.

7.3.4 Reduction of the model

Another method of studying the dynamic of the model is to directly investigate the trajectories of the different neurons. However, it is not possible to analyze and understand 200 neurons, each having its own trajectory. The number of neurons used by the model must thus be reduced to a small value.

The chosen value has been set to 20 neurons. Training the full model with such a small number of neurons to 58% accuracy is not possible as the task is too complex, there are not enough neurons to process all the different information in the sequence. The task needs to be adapted to have a model that can learn to the desired accuracy.

For this, a reduced task has been considered. This consists of simplifying the sequences such that a model with a small number of neurons can be trained. The sequences must still contain the characteristics of the serial recall task. For this, the lengths of the sequence will not be modified. Instead, the number of different items that can be used to form the sequences will be reduced. The smaller the number of different elements, the smaller the amount of different information the model will have to process.

However, the number of elements must also be high enough so that a large number of sequences can be formed with this number of elements, thus avoiding over-learning. To satisfy both conditions, 15 different items have been considered, modifying the input and output size of the model to this value.

Once the model trained, the neurons trajectories computed for the classical serial recall task, with a no-input period after it are represented in Figure 7.17

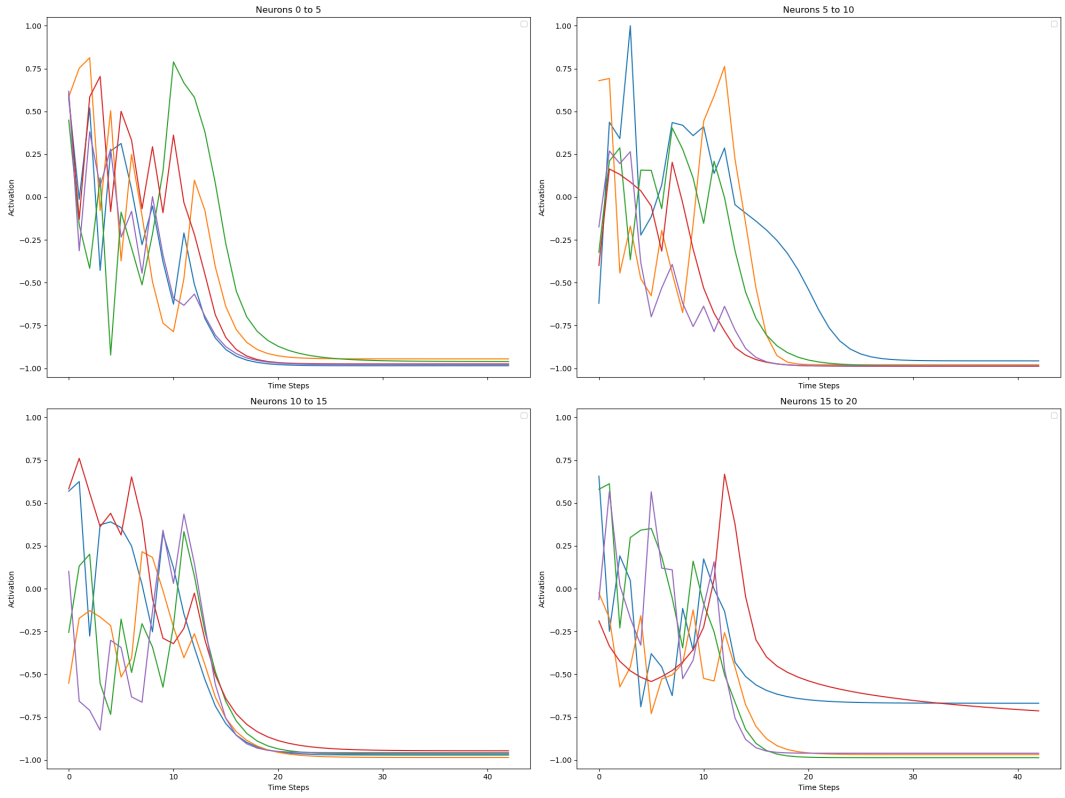


Figure 7.17: P.

During the task, all the activities vary greatly from one timestep to the other. There is always one neuron that is much more activated than the others, which could indicate that specific neurons encode either for specific timestep or for specific items. There are no clues of the use of bistability by the model, as the neurons does not maintain their last state over the timesteps.

Once the task is completed, almost all the neurons converge to -1. This means that all the neurons are in their most inhibited state possible. This was expected as they should not make any predictions during the rest period.

The same reduced model can be applied to the model using nBRC cells. The activity of neurons is represented in Figure 7.18.

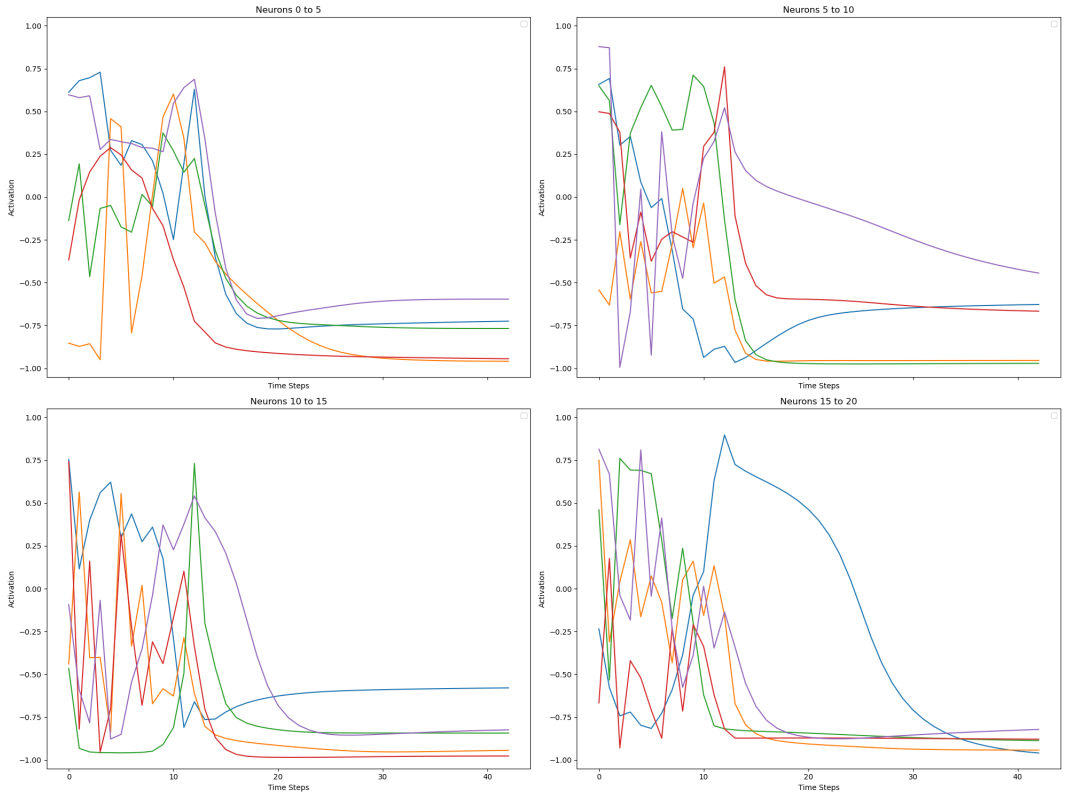


Figure 7.18: P.

The same overall behavior is observed for nBRC cells as for BRC cells. A difference is that during the no-input period, the neurons converge to different inhibited values, which could indicate that certain neurons may be playing a role in preventing the model from making predictions during this period.

With all the trajectories analyses, no evidences have been found on the use of the bistability property to solve the task. There are two potential explanations for this. The most simple is that the model does not need the bistability property to solve the task and therefore does not use it. This is highlighted by the fact that the Elman cell, which is monostable, can correctly learn the task.

The second potential explanation is that the task is too complex to detect the presence of the bistability by analyzing the trajectories. Indeed, the network has to process new information at each timestep. The neurons cannot maintain their last state as they have to always integrate new information.

Chapter 8

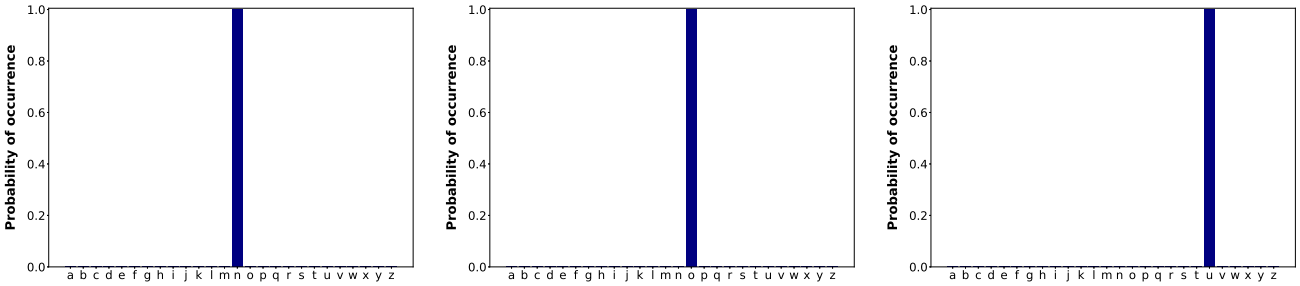
Exploring the model behavior

This chapter will focus on characteristics of the model and will try to assess their impact on its performances.

8.1 Probabilistic model

The output of the network, before making any prediction, is the probability distribution of the different letters, two different methods can be used, a deterministic or a probabilistic approach. On one hand, in the probabilistic approach, the predicted letter is chosen randomly from the probability distribution obtained after applying the softmax function to the output layer. On the other hand, the deterministic approach makes its prediction by choosing the letter with the highest probability.

One could discuss which of the two approaches is the most consistent for the model. For this, the output probabilities has been investigated when the network receives a sequence and correctly recall it. The probabilities during the encoding phase is represented in Figure 8.1. The sequence used as inputs is NOUGMJ.

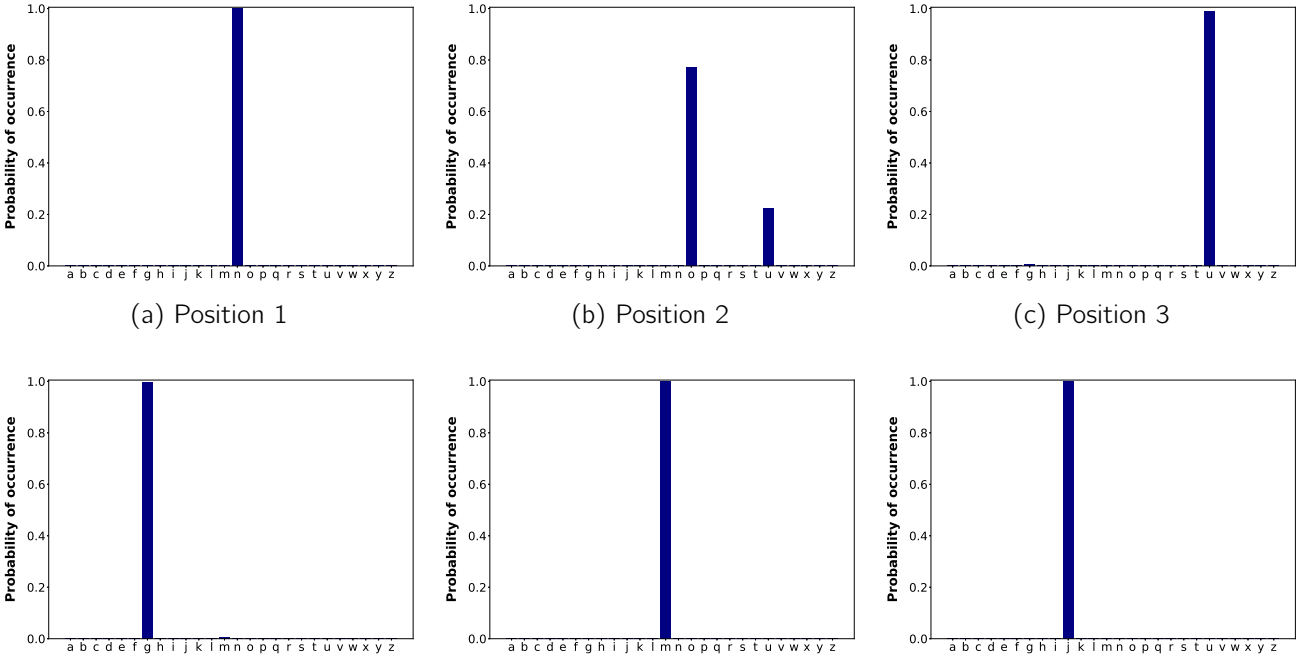


(a) Position 1 (b) Position 2 (c) Position 3

(d) Position four (e) Position five (f) Position six

Figure 8.1: Output probabilities of the network during encoding, for a correctly recalled sequence.

It can be seen that during the encoding phase, the output probabilities corresponding to the correct letter is always equal to one, and thus the others are equal to zero. This means that even with a probabilistic approach, the letters would always be correctly recalled during encoding. This was expected as the the task during encoding is really simple, activate the output node corresponding to the input node stimulated. For recall the task is more complicated. The output probabilities are represented in Figure 8.2.



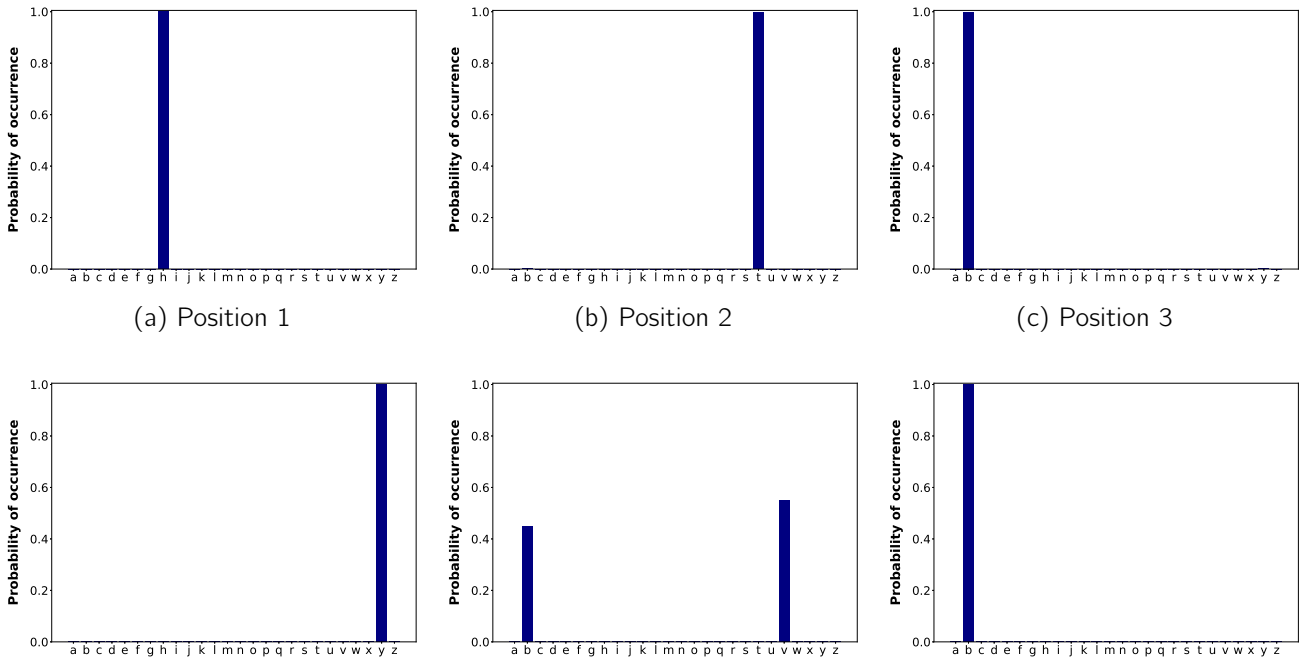
(a) Position 1 (b) Position 2 (c) Position 3

(d) Position four (e) Position five (f) Position six

Figure 8.2: Output probabilities of the network during recall, for a correctly recalled sequence.

It can be seen that, also during the recall phase, for the majority of the positions, the output probabilities are almost equal to one. This suggests that the model is really confident in its predictions. For the position two, it can be seen that an other letter than the correct one has a non zero probability. This is the next letter to predict. However, this probability remains small such that the model is still confident in its predictions.

One could argue that the output probabilities are that high because the network recalls correctly the sequence presented as input. To verify this, the ouptput probabilities during the recall phase of a sequence that is not correctly recalled by the network is represented in Figure 8.3. For this uncorrect recall, the model interchange the position of the last two items.



(a) Position 1 (b) Position 2 (c) Position 3

(d) Position four (e) Position five (f) Position six

Figure 8.3: Output probabilities of the network during recall, for a incorrectly recalled sequence.

It can be seen that when the recall is correct, the output probability is equal to one. When the model recall an incorrect item, the probability is much smaller, such that the network seems to hesitate between the two items but choose the wrong one.

In conclusion, using a probabilistic approach would not have a significant impact on the model as most of the output probabilities are equal to one. This has been confirmed experimentally, when no significant differences in the experimental effects have been found when using a probabilistic model.

8.2 Saturation of the network

Neural network saturation corresponds to a situation where new inputs are presented to the model, but the network is unable to learn any more. Its performance has reached a maximum and is no longer increasing.

8.2.1 Free saturation

During the training phase, the accuracy of the model increases as it learns the task. In the procedure used, training continues until a desired level of accuracy is reached, so that model performance matches experiments. However, it is interesting to assess the model behavior if we let the model learn as much as it wants: will it continue to learn until it achieves 100% accuracy, or will it reach a saturation point?

To characterize this, the model training is not stopped before one of these two conditions is reached. To characterize the model's behavior, the accuracy on the training set is shown in Figure 8.4.

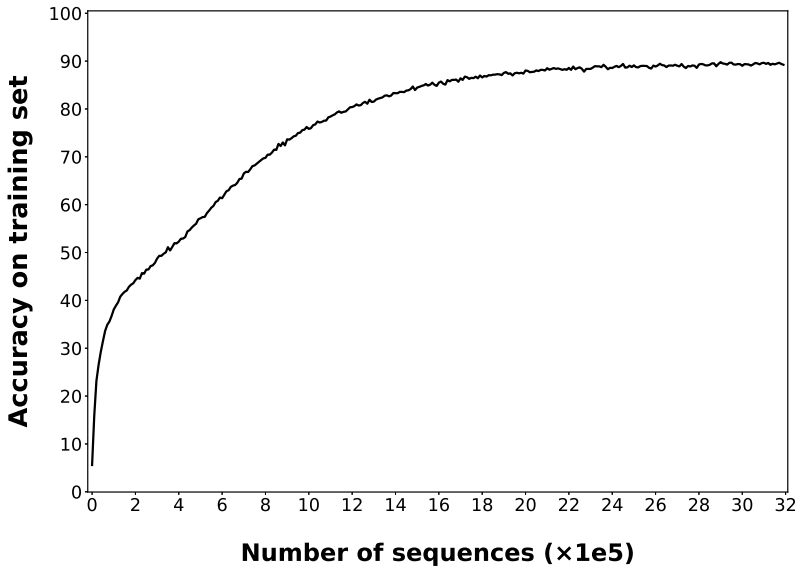


Figure 8.4: Accuracy of the network on the training set, i.e. sequences of length from one to nine, during the training of the model. The x-axis indicates the number of sequences presented to the network, while the y-axis indicates the accuracy in percent.

Before analyzing this curve, it is important to discuss why in this model it is relevant to consider the training curve of the model. In standard deep learning applications, the training set used for model training has a fixed size. Therefore, the model will better perform on them, as the model is updated to correctly predict these data. This means that considering the accuracy on the training set could overestimate the performance of the model.

As in the serial recall model developed, the training set is not fixed, new sequences are always presented to the network. This means that each sequence used during the training has a limited impact on the model performance, there will be no overfitting. For this reason, it is also relevant to analyze the accuracy on the training set in this model.

The accuracy values computed in this curve are the mean value of the accuracy on the 10 000 last sequences presented to the network. This curve is also the mean accuracy over sequence lengths from one to nine as sequence length varies from one sequence to another.

From the curve, it can be observed that the model has a very sharp increase of the training accuracy at the beginning of the training. As the network is initialized with random weights, modifying them in reaction to the computed loss will make the model learn rapidly at the beginning. This accuracy still increases after until saturation appears 240 000 cycles at around 88%. For larger number of cycles, the model does not learn anymore as the accuracy remains at the same level.

The same curve can be computed for a test set, composed of a fixed number of sequences of length six. If the model is indeed saturating, then accuracy on sequences of the same length should also saturate. This accuracy curve is represented in Figure 8.5.

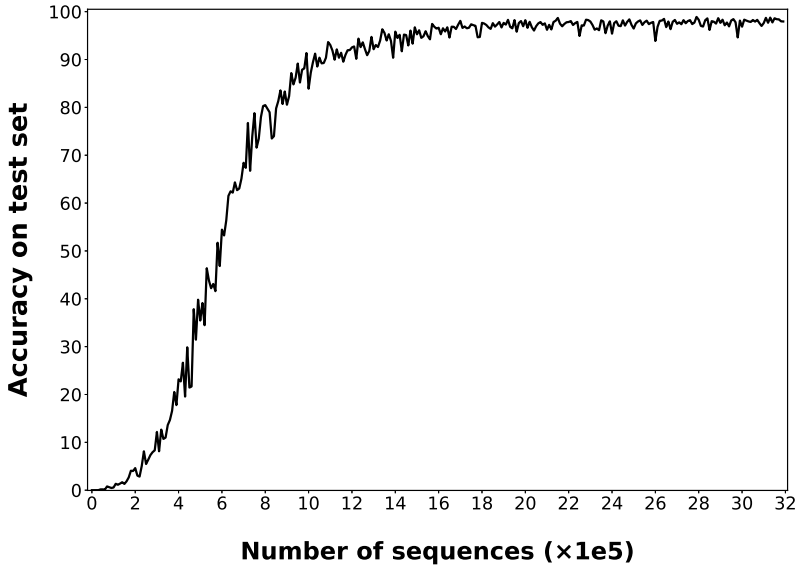


Figure 8.5: Accuracy of the network on the testing set, i.e. sequences of length six , through the training of the model. The x-axis indicates the number of sequences presented to the network, while the y-axis indicates the accuracy in percent.

Many different observations can be done on this new curve. The most important one is that the accuracy on this test set also saturates, which means that the model effectively has saturation accuracy and cannot perform any better than this level, even if the model is trained indefinitely. This is saturation, not perfect prediction by the model of all sequences of length six, since the average accuracy when the curve no longer varies is equal to 98%.

The saturation on sequences of length six is achieved more rapidly than the saturation of sequences of lengths from one to nine. This is because the model still not has finished to learn longer sequences as they are more complex.

The global shape of the test curve is also different from the training curve. The training curve has a monotonically increasing concave shape as the model is always learning, but better learn at the beginning for the reasons mentioned above. On the other hand, the test curve has a sigmoidal shape. This behavior can be explained by the fact that, at the beginning of training, the model learns more easily sequences of small lengths, as there is less information to be processed and remembered by the model. For increasing lengths, the sequences are more complex as they are composed of more items. It is less easy to remember all the items. Therefore, it will take more time for the model to learn how to recall them correctly. Once the model starts to learn this sequence length, the accuracy rapidly increases until saturation starts to appear.

Another difference is that the testing curve is more noisy. The main reason to this is that the test set is totally independent of the training procedure. Therefore, during the training, the model learns to solve the global task, not to correctly recall the sequences of the test set. This means that the accuracy should globally increases with the training, but could also present some variability from one evaluation to another.

Experimental curves at saturation

It is important to characterize the behavior of the model when it reaches saturation point. In fact, the model could still exhibit behaviors similar to those observed in humans, or it could behave in a completely different way.

To characterize this, the focus has been made on the two important experimental curves, which are the recall and the sequence-length curve. The curves of the saturation model will not exactly match the experiments as the model performance is better than when the model training is stopped after 58% accuracy is reached. The sequence-length of the saturation model is represented in Figure 8.6.

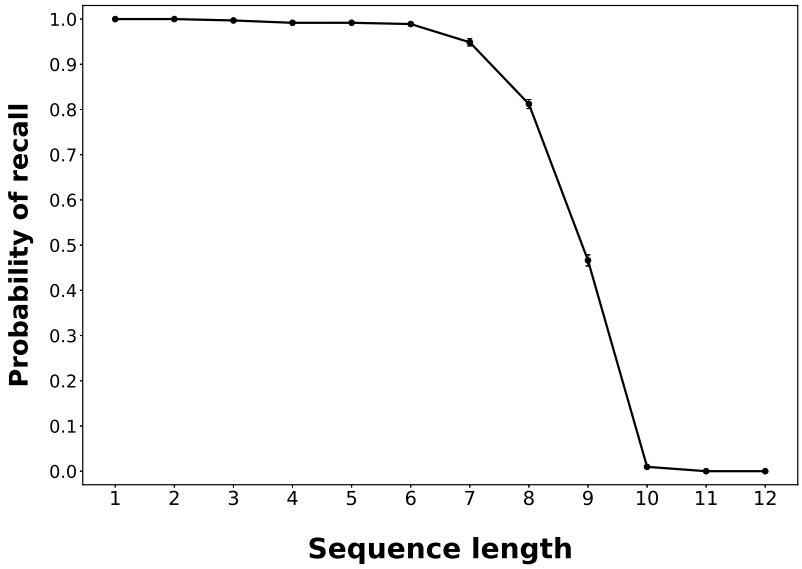
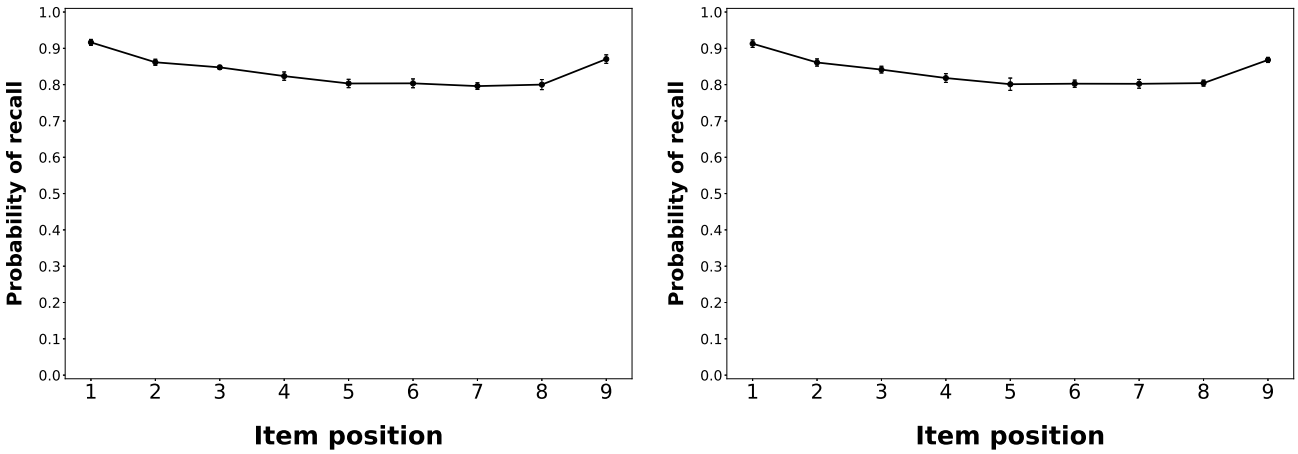


Figure 8.6: Evolution of the working memory performance with the length of the sequence for the model at saturation. The x-axis indicates the length of the sequence, while the y-axis indicates the probability of recall.

The sequence-length curve of the saturation model still has the typical shape observed in serial recall, i.e. the sigmoidal shape, but the curve obtained is also different. The memory span of the model is now equal to eight instead of six. All the sequence lengths below this value are almost perfectly predicted while the model performance for higher sequence lengths are almost equal to zero.

The sequence length showed that the recall probability of sequences composed of six items is almost equal to 100%. It thus does not make sense to look at its serial position recall curve as it will get almost 100% accuracy for all the item positions. Instead, longer sequences will be investigated, as they have a smaller recall probability. The serial position recall curves of sequences of lengths of eight and nine are represented in Figure 8.7.



(a) Sequences of eight items

(b) Sequences of nine items

Figure 8.7: Transposition gradients of the different cells.

The two recall curves present the primacy and recency effects, characteristic of the task. However, the recall probabilities are higher than the ones observed in the real experiments for these lengths. This is because the new model training allows the model to better learn the task and will thus have better recall performance than in humans.

In conclusion, letting the network learn as much as possible ultimately leads to model performance saturation. Once in saturation, the model still presents the typical experimental shapes, but shows better recall performances.

The same investigation has been done on the BRC network. The model also presents a saturation after its intensive training. However, the recall performance at saturation are better in the BRC network than in the Elman, even if the training procedure and the hyper-parameters of the network are identical. This means that increasing the complexity of the network dynamic enhances the model performance.

8.2.2 Tuning the saturation

In the last section, it has been shown that the network reaches a saturation during its training. In this section, we will try to tune these saturation performances, i.e. modify the network such that its performance saturates at a desired accuracy value.

One could ask why it is important to be able to modify this saturation value. One major limitation of the developed serial recall model comes from the stopping criterion used to stop the model learning. Indeed, this method is a bit artificial. The training is stopped when the desired accuracy level is reached, such that the model performances are forced to a given value instead of being an inherent property of the model.

Moreover, this stopping criterion is also not really plausible in humans. The brain does not stop learning the task once a given accuracy is reached. Instead, it continuously learns and assimilates new information, but reaches a saturation, where its performances on the specific task cannot increase anymore. This is why being able to tune the saturation of the model is important. This will allow to create a model having performances that naturally saturate to the values observed experimentally, making this characteristic an intrinsic property of the model.

To tune the saturation, the structure of the model and its training cannot be modified, otherwise it would result in a totally different model, with potentially different experimental results. In standard deep

learning approach, hyper parameters play a central role to modify the performance of the network. These parameters will also be investigated in this case.

In the serial recall model, three hyper parameters have been used: the batch size, the learning rate, and the number of neurons. The batch size will not be changed, as its value has been chosen to one to be neuronally plausible. The learning rate is a very sensitive parameter, meaning that very small changes of its value will have huge impact on the model saturation performance, which does not allow fine-tuning of this value. The last parameter is the number of neurons in the hidden layer used to process the information in the network.

Reducing the number of neurons leads to a lower saturation level, as desired. This means that depending on the value of this parameter, the performance of the model will be different. It is now important to find a value such that the model saturates on 58%, which is the mean performance observed in humans. For this, this hyper-parameter has been gradually decreased until the model saturates to a value close to this one.

To reach this value, 55 neurons have been used. When searching for this value, it has been found that there is a wide range of values where the saturation level only slightly decreases as the number of neurons decreases. This is a region of small sensitivity, i.e. modifying this parameter has a small impact on the model saturation performance. For much lower values of this parameter, i.e. below 80 neurons, this parameter has a much higher sensitivity on the model performance, i.e. modifying this parameter will modify the saturation level. The testing accuracy for the model using 55 neurons is represented in Figure 8.8.

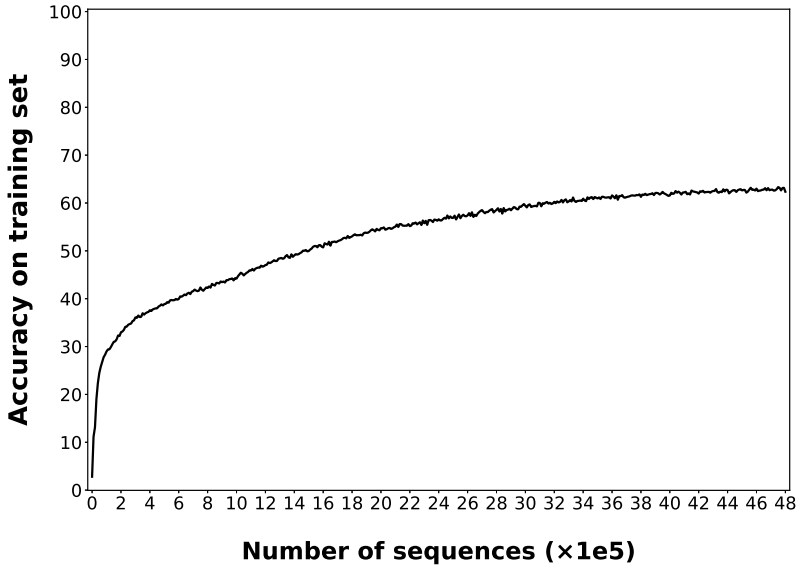


Figure 8.8: Accuracy of the network on the testing set, i.e. sequences of length six , through the training of the model. The x-axis indicates the number of sequences presented to the network, while the y-axis indicates the accuracy in percent.

It can be seen that the accuracy on sequences of length six saturates at approximately 58%. Even with this saturation, variations of the accuracy are present. This indicates that the same network architecture can present some variability, but this will be explored in the next chapter. To check the saturation of the model, the accuracy on the training set is represented in Figure 8.9.

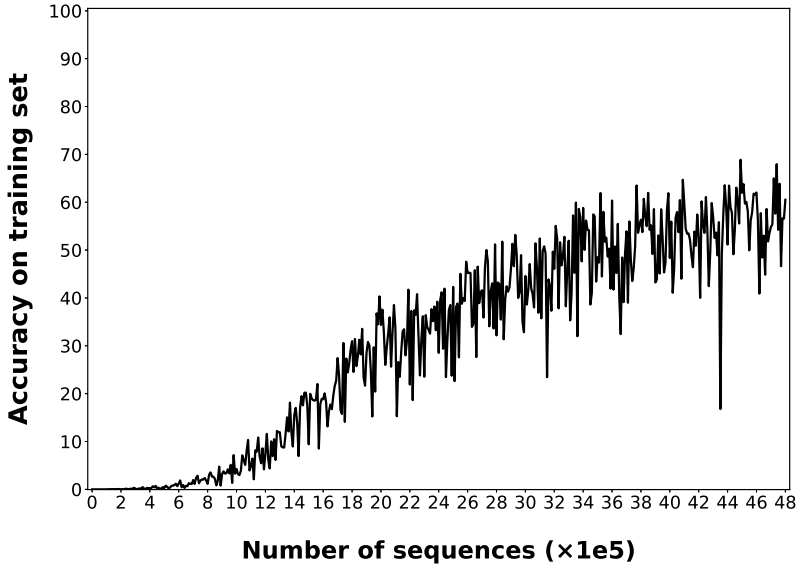
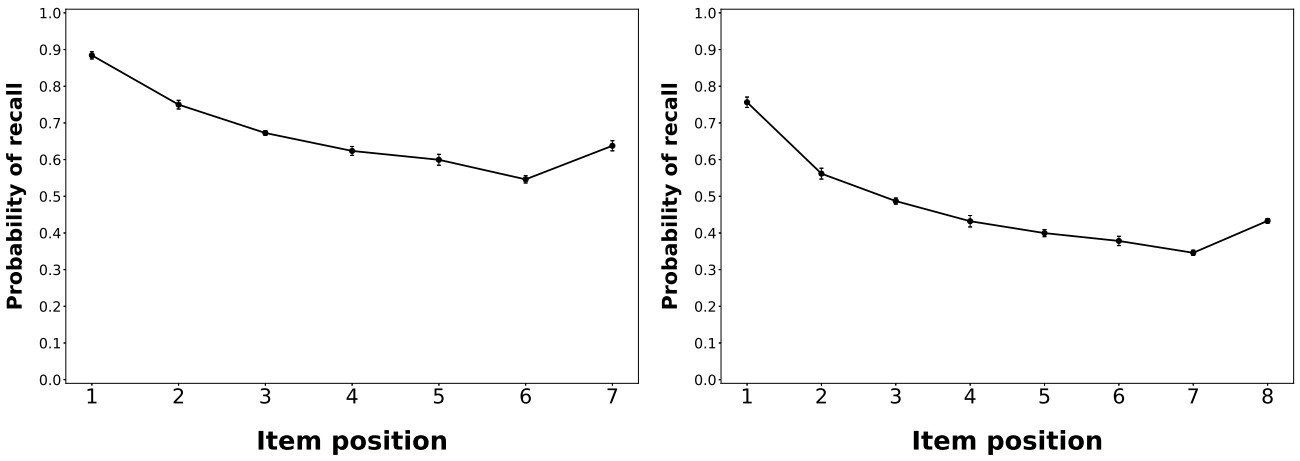


Figure 8.9: Accuracy of the network on the training set, i.e. sequences of length from one to nine, during the training of the model. The x-axis indicates the number of sequences presented to the network, while the y-axis indicates the accuracy in percent.

This figure shows that the whole model performance also saturates once sufficiently trained. It has been shown that the free saturation model has the typical characteristics of the shapes of the experiments. It is now important to assess that the model saturating at 58% has exactly the same behavior as in humans and the model trained using the stopping criterion. The sequence-length curve and the recall curve on sequence of length six are represented in Figure 8.10.



(a) List length of seven

(b) List length of eight

Figure 8.10: Serial position recall curve for the tuned model.

The two curves predicted by this new model are very similar to the experiments and the previous model. With this approach, it has thus been possible to create a model that naturally has the same performance as in humans. This performances are intrinsic to the model contrary to the previous model, where stopping the training is more artificial.

8.2.3 Mapping the saturation and number of neurons

All the participants do not have the same performances on the task. Indeed, some perform the task better than the others. It is therefore also important to be able to represent other recall performances with the model. As discovered in the last section, the performances at the saturation of the model can be tuned by changing the number of neurons in the model. By using this, we characterized a mapping between the recall performances and the model architecture. This mapping is represented in Table 8.1.

Number of neurons	**Mean accuracy on sequences of length 1-9**	**Mean accuracy on sequences of length 6**	**Standard deviation on sequences of length 6**
50	56.35%	34%	4.9%
51	59.25%	34%	4.9
52	60%	41%	5.6
53	61.2%	49%	7
54	62.4%	50.3%	6.36
55	63%	55.4%	5.3
56	64.23%	57%	6
57	65.23%	60.4%	5.2
58	66.47%	64%	7
59	68%	71%	5.1
60	70%	75%	5

Table 8.1: Mapping between the model performances and the number of neurons.

This mapping shows that, with a small range of values for the number of neurons, the model can cover all the plausible recall performance values of humans. The increase of the mean performances of the model is monotonic, meaning that the higher the number of neurons in the model, the better its performance.

The increase in the mean recall performances on sequences of length six seems to be linear. To check this, a linear regression has been applied to the recorded data, which gave $R^2 = 0.98$. This confirms that in the range of values considered, an increase in the number of neurons corresponds to a linear increase in the model performance.

The standard deviation of the performance is approximately the same for all the number of neurons. This indicates that even if the model saturates, This characteristic is essential as it allows the model to cover all the possible recall performances and not just some discrete values.

Finally, we might have expected the number of sequences required to saturate the network to decrease with the number of neurons, as the information is processed by a greater number of neurons and the task can therefore be learned more easily. However, this was not observed in the experiment.

This mapping allows to better represent different participants. Indeed, all humans have different brains, whether in terms of connections between neurons or slight structural differences. These differences in structure correspond to different model architectures, characterized by different numbers of neurons. Therefore, this mapping allows to obtain more plausible models.

8.3 Model generalization

Until now, it has been shown that the model developed is able to correctly recall sequences composed of familiar items, i.e. items on which it has been trained on, as in real experiments. However, one could ask if this model is also able to generalize and correctly recall sequences composed of new items, i.e. items that were never used to train the model.

To test this, the input layer has been divided into two subsets. The first subset was composed of 16 input neurons and was used to create the sequences fed during the training of the model while the 10 remaining input neurons were used to compose the new unfamiliar sequences on which the model will be evaluated. This means that the sequences of the training and the testing phase are completely different.

For the model architecture and training procedure, absolutely nothing has changed, as we wanted to test the ability of the network to generalize on new items. The cell used in the model was the BRC and network has been completely trained after 8 000 cycles, as in the training of the other simulations.

The model obtained unable to recall the new sequences correctly. This result does not match at all with the real experiments. However, this behavior was expected. Indeed, the new items used only during the testing corresponds to new neurons in the input layer. As they have not been involved during the training process, their connections with the neurons in the hidden layer have not been optimized at all such that they can be used to correctly recall the items. Consequently, when a new input is provided to the network, the model will predict a item on which it has been trained on, which is what is observed in the model's outputs.

In humans, things happen differently. Indeed, when unfamiliar items are presented to the participants, they still manage to recall the items. However, the recall probabilities in are much smaller and they do much more omissions.

8.4 Model Variability

In all the previous simulations, a very large set of sequences has been used to reproduce the different psychological effects, leading to the obtainment of mean curves. However, in real experiments, variability is present in the experimental data and is the result of many different factors. In the different models developed over the years, this characteristic has rarely been studied. This section will explore the different sources of variability of the model and assess their impact on the experimental curves predicted by the model. For this, a focus has been placed on two different curves: the working memory performance as a function of the sequence length and the serial position recall curve.

In the literature, two main measures are used to compute the variability of the curves. The most used is the standard deviation, which characterizes how the measured values deviate from the mean. The second measure is to create a 95% confidence interval. The expression of this interval is the following.

$$\left[\bar{Y} - Q_{t_{n-1}}(1-\alpha/2)\frac{S}{\sqrt{n}}, \bar{Y} + Q_{t_{n-1}}(1-\alpha/2)\frac{S}{\sqrt{n}}\right]$$

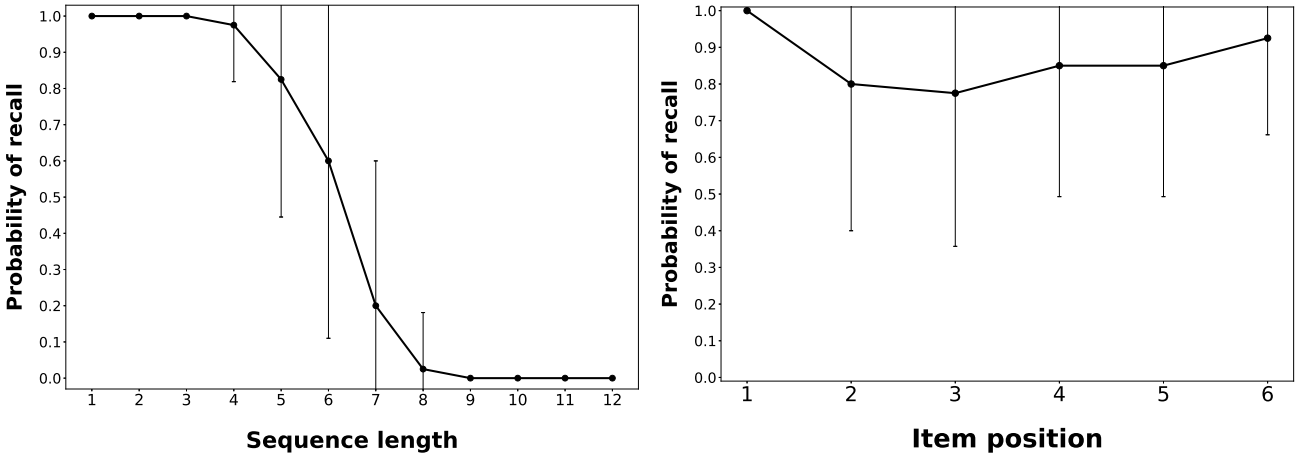
Both options have been investigated. As they give similar results and the standard deviation is most used in literature, only the last measurements will be used to present the results in the next sections.

8.4.1 Sequences variability

The first source of variability comes from the different sequences presented to the same model. Indeed previously, to test the model, 10 000 sequences have been used to create the experimental curves. This

huge number ultimately results in a small variability as the standard deviation depends of one over the number of sequences.

In humans, this psychological task is not done on such a large number of sequences, this would not make any sense as this would take too much time, and sufficient results can be acquired with a quite small number of sequences. The typical number of sequences presented to a participant doing the task is around ten. Therefore, the testing procedure has been adapted to fit the experiments. The experimental curves obtained with this new procedure are represented in Figure 8.11.



(a) Evolution of working memory performance with sequence length

(b) Serial position recall curve

Figure 8.11: Sequences variability of the experimental curves

It can be observed that the two curves present some variability. For the curve of the working memory performances as a function of the sequence length, the sequence lengths close to the memory span present the highest variability. This is because for small lengths, all the models can accurately predict the majority of the sequences, while for large sequences, no model can correctly predict the sequences. For intermediate lengths, the model has the most variability as it correctly predicts only a part of the sequences.

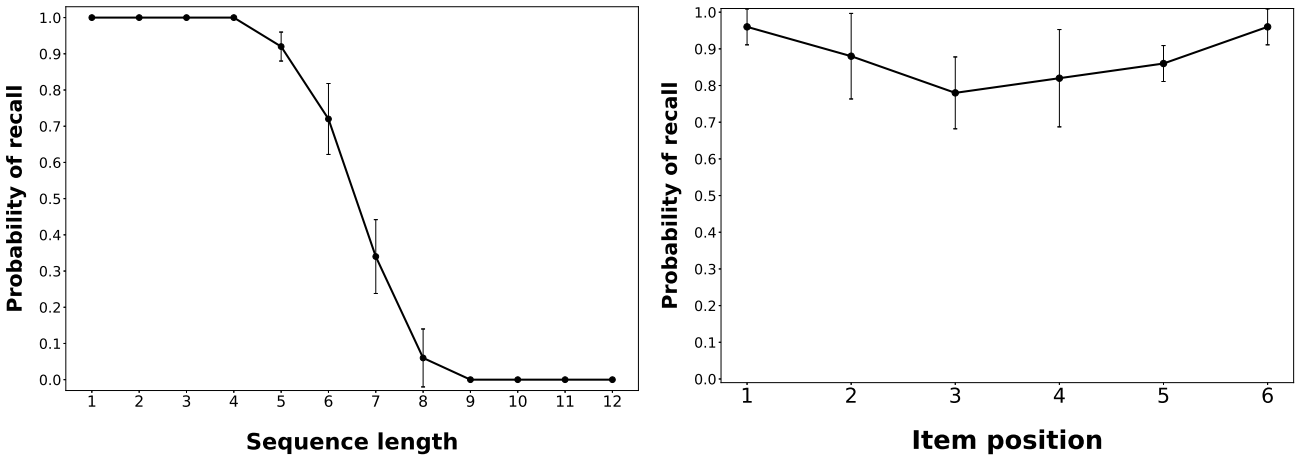
The serial position recall curve also presents variability, which is more significant for the items in the middle of the sequences than at its extremities. This is because the model will always better predicts items at the beginning and the end of the sequences due to the primacy and recency effect. Therefore, it has a small variability, whereas the recall performance of the items in the middle of the list will depend on the list presented.

However, the variability obtained is too important compared to what is observed experimentally. This is because in this configuration, each item is considered as a binary output, either correctly recalled and is equal ton one or not and is equal to zero. This is reduced when the mean accuracy is computed but this high variations are captured by the standard deviation, explaining these high values. As this variability is too important, we need to find a way to reduce it while keeping this characteristic.

A second type of experiment performed in psychological study is to characterize how the mean performances of the participants change over several sessions. For this, several sequences are presented to the participant at each session and the evolution of the mean performance over the sessions is computed.

To reproduce this, five sets of ten different sequences are created. For each set, corresponding to a session, the mean performances are computed. Then, the experimental curves are computed based on

the performance curves obtained during the different sessions. The computed results are represented in Figure 8.12.



(a) Evolution of working memory performance with sequence length

(b) Serial position recall curve

Figure 8.12: Sequences variability of the experimental curves

The positions in the two curves where the variability is the highest are exactly the same as in the first simulation. This was expected as it is a property of the model to have most variations in predictions at the memory span and for items in the middle of the sequences. In addition, the variability between different sessions is less important than the variability of a single session, as shown in the first simulation. This is because when looking at the variability across multiple sessions, the different mean curves computed are compared. These curves are more similar and thus will present much more plausible variability values, than comparing the recall or not of items, which leads to binary outputs.

When the number of sessions or the number of sequences per session is increased, the variability is further reduced as, when there are more sequences per session, the mean performances for each session will be more similar or because, when there are more sessions, variability is also reduced as there are more data.

As a result, the number of sequences on which the model is tested is an essential parameter to produce variability. When a small number of sequences is used as in the experiments, a high variability of curves can be obtained as observed in the experimental task.

8.4.2 Training variability

A second type of variability is the training variability, which comes directly from the training of the network. Indeed, training the model multiple times using the same architecture and training procedure will result in different final models.

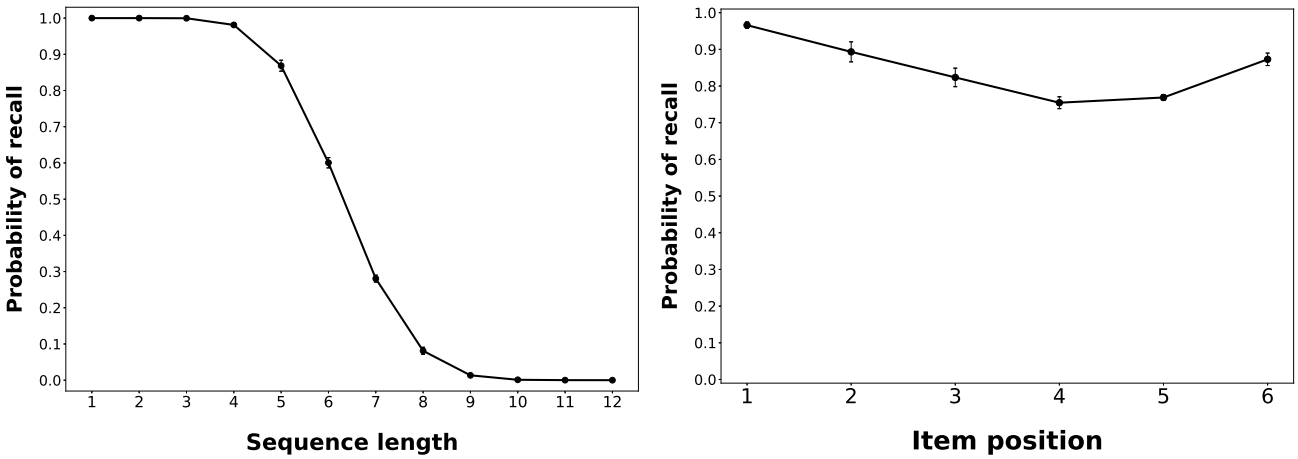
This training variability can be identified as a first source of variability between different participants. Indeed, different training leads to different models, even if they are only slightly different. These models can be associated with different participants, as none of the human brains are identical. It is therefore not cognitively plausible to use exactly the same model to represent different individuals. This variability in training is the consequence of many different reasons.

The first reason is that the network weights, used to shape the network connectivity, are initially randomized. This means that even before starting the learning procedure, the initial states of the different models are a bit different. This will ultimately lead to different final models.

The second reason is that, as the dataset used to train the model is not fixed, the sequences used to train the different models will be different. Even if the model is not subject to overfitting, the sequences presented to the network will still influence how the model learns and its final weights.

Finally, the last reason for these differences is the criterion used to stop the model training. Indeed, even if the same value of the criterion is used for the different models, it is not possible to stop the training exactly at this value. Moreover, the test set used to compute the accuracy varies between the model training. Both of these elements add variability on when the model training is stopped.

When all combined together, these reasons will lead to different models with the same general behavior, but with small variations. These variations due to model training are represented in Figure 8.13.



(a) List length of seven

(b) List length of eight

Figure 8.13: Transposition gradients of the different cells.

As expected, slight variability can be observed in the curves. However, it is much less important than for the sequence variability. This is because the different sources of variations explained above only lead to slight differences between the different models. Therefore, the training cannot be considered as a significant source of variability.

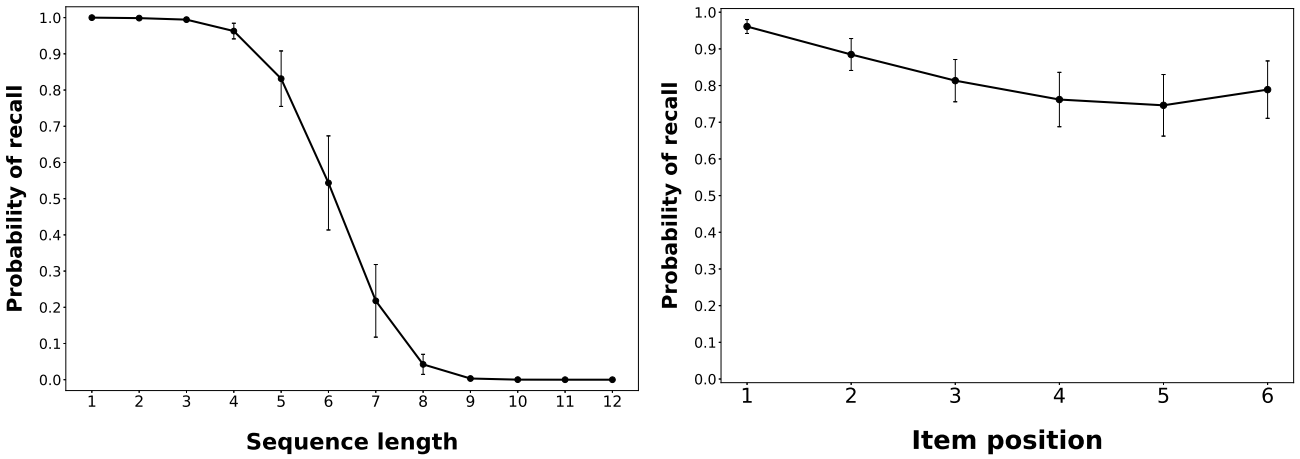
8.4.3 Model variability

Until now, the models have been trained such that the recall performances on sequences of length six reached 58%, which is the mean performance recorded in some studies. This does not mean that all the participants have the same performances. Indeed, some people will better perform the task while others will have worse performances, this is inter-participant variability.

To represent this variability, the stopping criterion of the training will be modified such that it will be different for each model. Before carrying out the experiments, we need to define how to select the value of the stopping criterion. There are no studies that define how the working memory performance on sequences of length six varies with individuals. It is thus not possible to be as close as possible to the real experiments.

Instead of this, a small experiment has been considered to show the importance of inter-participant variability. For this, the stopping criterion is drawn from a uniform distribution. The lowest bound of the distribution has been set to 35%, as it has been assumed that only the participants with a minimal performance are kept in the studies, while the highest bound has been set to 85%, as it has been assumed that there is an upper bound on the real performances of the participants.

By using these procedures, the accuracies used to create the inter-participant variability curves are 50.48, 57.7, 47.3, 61.3, 74.5, 58 and 41%. The two curves are represented in Figure 8.14.



(a) Working memory performance with sequence length

(b) Serial position recall curve

Figure 8.14: Variability of the experimental curves of Elman cells, using model variability.

These plots show high variability, highlighting the importance of different participants in the results. The variability obtained in these curves is dependent of two different factors: the number of participants and the different accuracy values.

For the number of participants, the higher its value, the lower the variability, as the curves become more and more averaged. For the stopping criterion of the models, depending of the accuracy values drawn, the variability will be more or less important in the curves. Indeed, if all the accuracies are close to each other, then this will result to small accuracy, while bigger differences of the accuracies results in bigger variability. This is in line with the real experiments, i.e. if the participants have similar recall performances on the presented sequences, the recall curves will have a small variability.

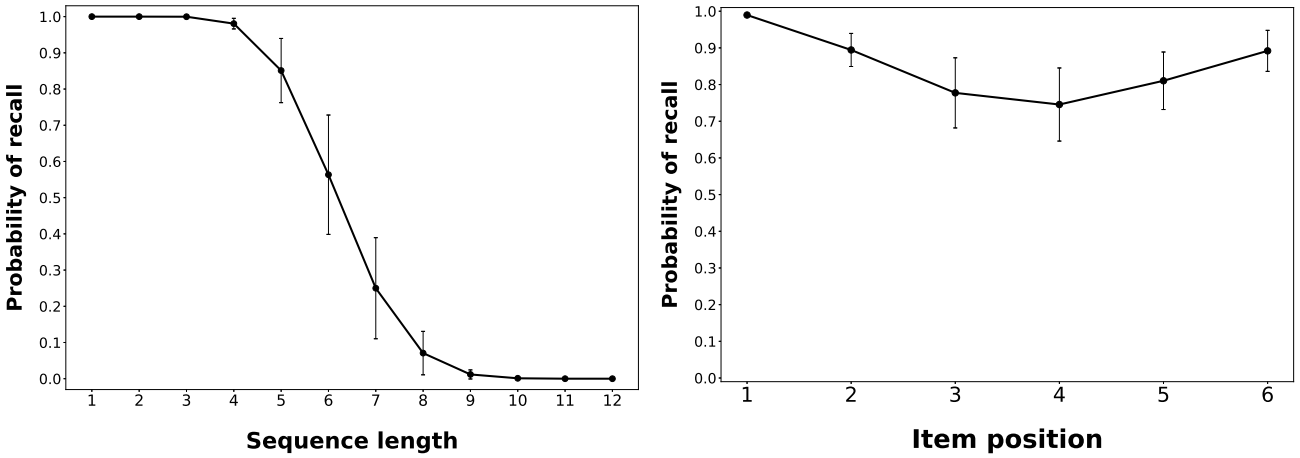
8.5 Combining model saturation and variability

The two last concepts, i.e. variability and model saturation, can be combined together to form a more plausible model.

First, the training variability can be combined with the performance saturation of the model described in the previous section. The introduction of this model saturation removes the variability coming from the stopping criterion, but introduces a new source variability. Indeed, when the network saturates, the accuracy on the sequences of the same length varies a lot from one epoch to the other, leading to different final recall performances depending on when the model learning phase is stopped. This will create a higher variability of the curves, but this effect will be explored in more detail with the model variability.

For the inter-patient variability, the mapping between the number of neurons and the model saturation performance can be used. This allows to represent different participants with different model architectures and so better represent the difference between them. To use this mapping, the recall performance is still drawn from the uniform distribution. Then, the mapping is used to find to which number of neurons this performance corresponds to and the model is finally trained with this architecture.

To show that the variability results are also valid for the BRC model, the experiments have been done using this cell. The curves obtained using the mapping between the number of neurons and model saturation are represented in Figure 8.15.



(a) Working memory performance with sequence length

(b) Serial position recall curve

Figure 8.15: Variability of the experimental curves of BRC cells, using model variability in combination with the model saturation mapping.

8.6 Conclusion

This chapter studied fundamental characteristics of the model to better understand its behavior. By introducing the saturation, we achieved to translate the stopping criterion of the training into an intrinsic property of the model, making it more plausible. Moreover, with the addition of the variability of the curves, the model became even closer to what happens in real experiments.

Part IV

Model limitation, conclusion and perspectives

Limitations of the model

Despite the many effects the model is capable of reproducing, the model still present some limitations that will be discussed in this section.

First, the model is not able to reproduce the main confusability effects observed in experiments. When the sequences are composed of only confusable or non-confusable items, the curves are similar to the experimental curves, with better performances on the sequences with only non-confusable items. However, when both confusable and non-confusable items are alternated, the serial position recall curves slightly present the typical sawtooth pattern, but are much less pronounced than the real curves. The confusability effect is thus less important in the model.

Then, the model cannot handle new items. If the sequences are made up of items that have never been presented to the network during its training, the model will never recall these items. The model is bad at generalization.

The model also struggles to generate large transposition errors, which is also important in the real experiments.

In conclusion, there are still different points to investigate in order to assess if a recurrent neural networks can be used or not to reproduce the effects described abive and others that have not been investigated in this work.

Conclusions and perspectives

Despite the fact that recurrent neural networks have rarely been studied as a model for the serial recall task, they have shown great potential to represent it. Indeed, the model was able to represent many of the effects observed in serial recall: bow-shaped serial position recall curve, transposition gradients, working memory performance as function of sequence lengths,... This is important as many struggled to recreate a recurrent neural network that is able to reproduce these results.

By extending the model, other experimental effects were studied in relation to Botvinick and Plaut's original model, such as different types of error, backward recall, etc. The comparison of different cell dynamics highlighted that the BRC cell, which is neurally more plausible, allows to obtain predictions closer to the experiments. However, this was not the case for the nBRC cell, which presented an asymmetric version of the serial position recall curve.

Finally, the model has been modified to be more neurally plausible, by introducing model saturation so that the model recall performance is an intrinsic property of the model. In addition, variability, which is an essential characteristic of the experiments, has been implemented and its different sources have been discussed. This is even more important because this characteristic is rarely taken into account in the various models developed.

As this model showed great potential, different perspectives can be considered for the model to further investigate its behavior.

A first perspective is to investigate the complex span version of the serial recall task. This task consists of presenting a processing task between the presentation of the sequence and its recall. The processing task can be reading sequences, doing arithmetic, etc (*Bayliss et al., 2003*). This task is more complex, as the participants must remember the elements of the list while performing the other task. Therefore, the recall performance will be different. Many investigations have been conducted on this task and the mechanisms involved. Recurrent neural networks could give a new point of view on it.

The effect of the presentation rate could also be assessed with this model. This effect highlighted better recall performances when the items are presented for a longer period to the participants. This could be assessed by using a saturation model with the same architecture. Then, the number of timesteps where the items are presented to get longer inputs. Finally, the saturation performances for different duration can be compared to assess if longer duration of presentation leads in better performance in the model, as in human benchmarks.

A final prospect is to study all the different experimental effects of this work, with an even more brain-inspired cell, the recurrent spiked cell (*Geeter et al., 2024*). This cell was created by modifying the BRC cell used in this work by introducing slow negative feedback. This creates spike-based activity in the network, similar to what happens in real neurons in the brain.

Appendix A

Exploring delay in the task

A.1 Introduction

The bistability property of biologically inspired cells did not seem to be a necessary mechanism to represent the immediate serial recall task. To investigate the impact of this property, a simplified version of the serial recall with delay task will be studied.

This variation of the task consists of presenting a delay period between the presentation of the sequence and its recall. During this delay, the participant should typically perform a completely different task, used as a distractor, such that the participant's mind is focused on a task other than serial recall.

However, to better characterize the bistability of the cells and how they keep the information in memory, a simpler version of the delay has been used instead of the distractors. This simplification consists of using a period of emptiness, corresponding to a period where no inputs are presented to the network, during the delay. This period of no inputs also implies the suppression of the output feedback, such that the neurons of the hidden layer have to maintain the information on their own.

The experimental curves seems to be not affect by the presence of this delay (*Ricker et al., 2016*;*Ricker et al., 2020*) .
A perspective for the model is to implement the more sophisticated version of the delay, using distractors, to assess the ability of such model to reproduce these experimental results.

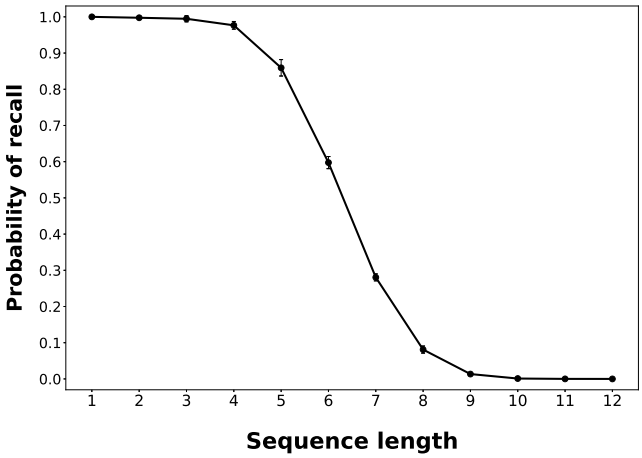
A.2 Serial position recall curve

To explore this new simulation, the dataset has been modified by adding the no input period between the learning and recall phases of the sequences. During this period, the network can do whatever it wants, its predictions are not taken into account in the loss and therefore will not influence the update of the model. These modifications are the only ones necessary to learn the new task.

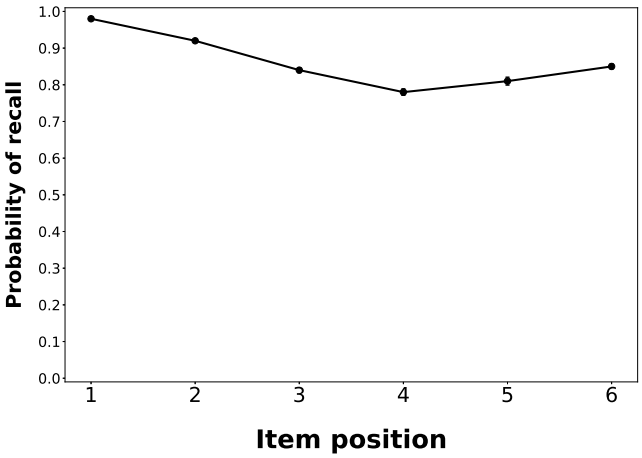
The task has been learned with a fixed delay period, and then its generalization has been tested on other delay periods. The delay value used to train the model is equal to ten. First, the model was trained using the Elman cells. No matter the number of sequences presented to the network during its training, it is not able to reach the approximate accuracy of 58%, seen in experiments. This can be explained by the fact that the Elman cells have a fading memory. They can keep information in memory for a short period of time, but struggle to maintain them at long term, explaining why they cannot correctly recall the sequences after the delay.

Training the model on the task without any delay and testing its performance on the delay task have also been done. However, in this case, the model was not able to understand the delay and take it into account, regardless of the cell used. This was expected, as the model has never been trained on this, it does not know what it means and what it is supposed to do during it.

The focus in the rest of the chapter will be on the two bistable cells, the BRC and the nBRC. The serial position recall curves are represented in Figure A.1.



(a) Evolution of working memory performance with sequence length

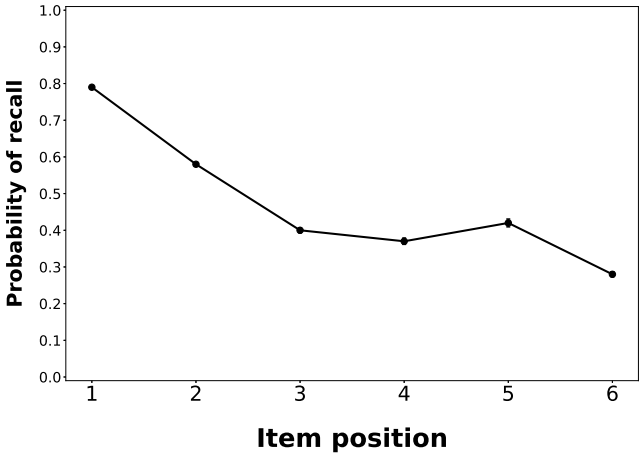


(b) List length of eight

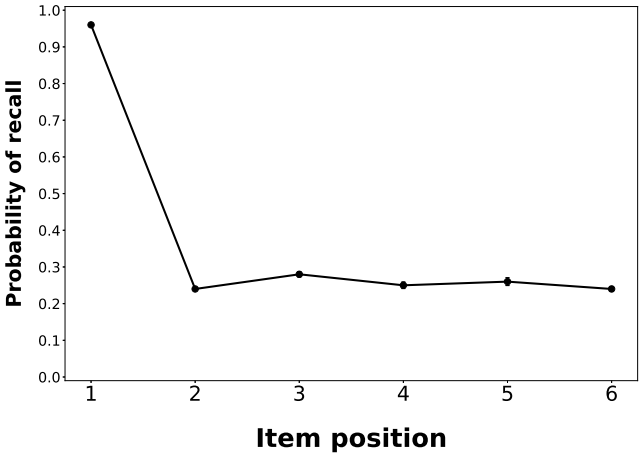
Figure A.1: Experimental curves for model trained on a delay of 10 timesteps without teacher forcing.

During model training, it was not possible to reach the target value of 58% of accuracy using the BRC cells with a delay of ten. The maximum recall performance obtained was around 52% and the learning phase has been stopped at this value. This decrease in accuracy in the training could be in line with the fact that using a delay slightly decreases the performance on the task.

When looking at the shape of the curves, it can be seen that they still have the characteristic shape of the recall curve, meaning that the delay does not affect the global recall curve shape, which is in line with the experiments. The generalization of the model on different delays has to be assessed to check if the model is able to represent the delay task correctly. This generalization is represented in Figure A.2.



(a) Delay of 5 timesteps

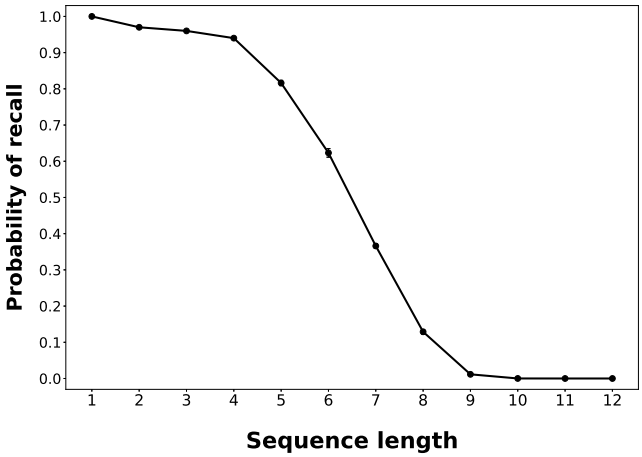


(b) Delay of 15 timesteps

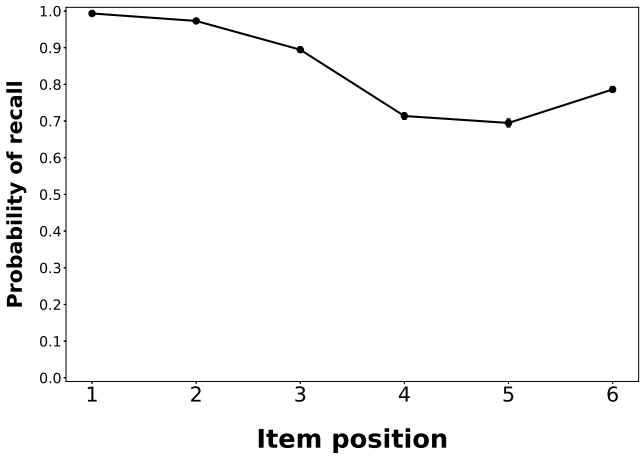
Figure A.2: Transposition gradients of the different cells.

The recall curves of the BRC model completely fall apart whether the delay is reduced or increased. For both delays, the recall performances do not present the bow-shaped curve. The model is not able to remember the items correctly. In real experiments, the delay only slightly affects the recall curves. For smaller delays, the recency effect is no longer present and the performance are worst. This is not wanted as when reducing the delay, the curves should be closer the ones obtained without any delay.

This means that the BRC cell is not able to remember the different items by maintaining their information on delays for which it has not been trained. The same experiments have been applied on nBRC cells to investigate if this leads to better results. The recall curves of the nBRC model on a delay of ten is represented in Figure A.3.



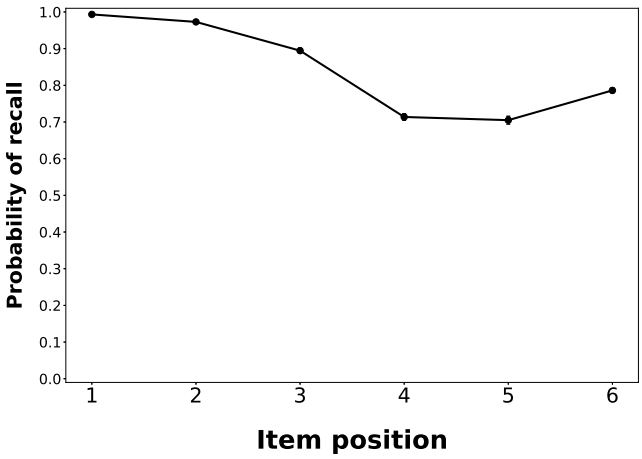
(a) Evolution of working performance with sequence length



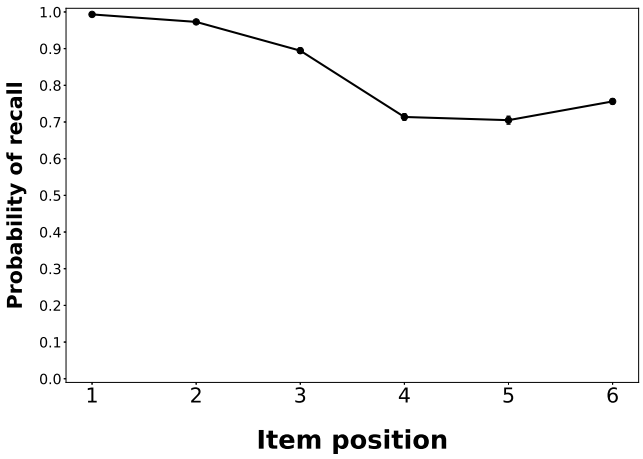
(b) Serial position recall curve

Figure A.3: Experimental curves for the task with delay of 10 timesteps.

The recall curves have the typical shape of the serial recall task and the evolution of working performance also have the same global shape expect at the beginning. nBRC cells seems to correctly handle the delay. To check this, the generalization on other delays has been assessed in Figure A.4.



(a) Delay of 15 timesteps



(b) Delay of 100 timesteps

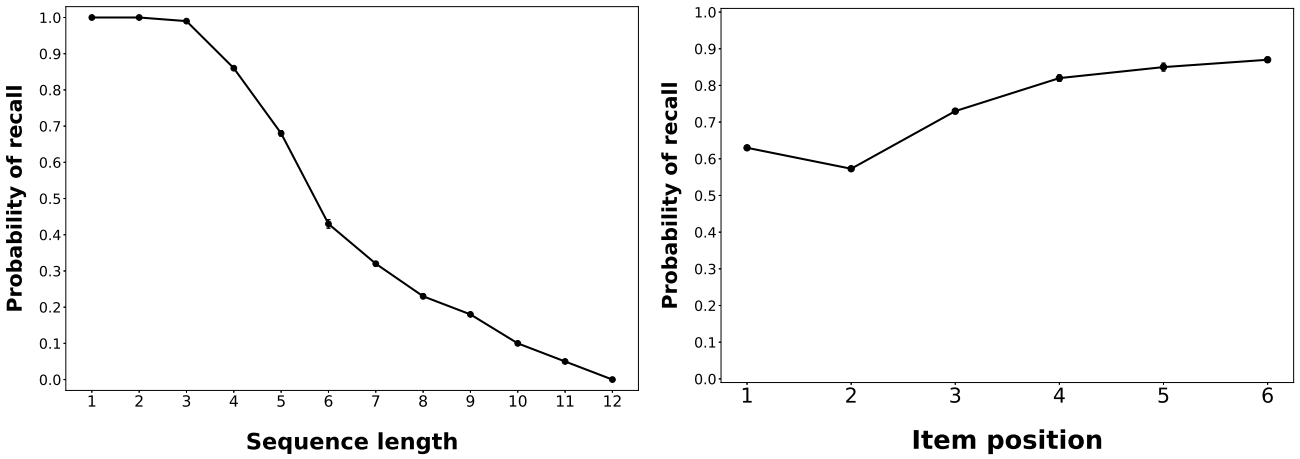
Figure A.4: Serial position recall curves of nBRC cells for different delays.

The curves are only slightly affected by the presence of the delay, which is in line with the data (*Ricker et al., 2020*). However, if the nBRC is capable of handling the delay, it is not capable of producing the

correct shapes for the serial position recall curve.

Effect of teacher forcing

The effect of teacher forcing on the delay task will be characterized. It has been discussed in a previous chapter that removing teacher forcing was important to obtain proportion of errors predicted by the model closer to the experiments. This chapter will present a second reason against teacher forcing. For this, the model using BRC has been trained on a delay of ten. The serial position recall curves of sequence lengths of six and eight are represented in Figure A.5.

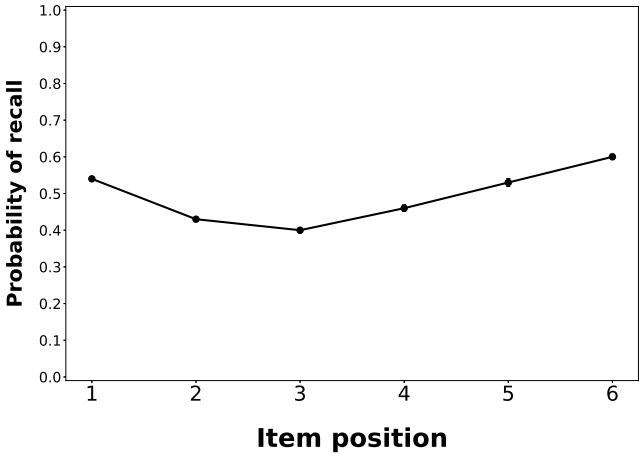


(a) Evolution of working memory performance with sequence length

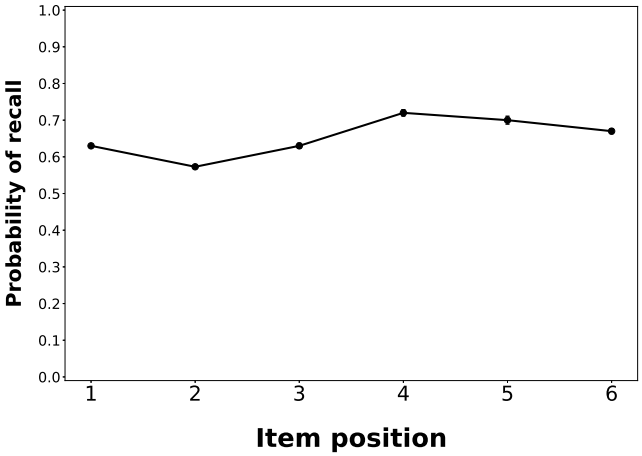
(b) Serial position recall curve

Figure A.5: Experimental curves of a BRC model trained on a delay of 10, without using teacher forcing.

The serial position recall curves are totally different from the curves without delay, the primacy effect is much smaller than the recency effect, and the recall probability increases with the item position in the sequence. This means that the model better recalls the last items than the firsts. Moreover, the curve for sequences of six items is highly asymmetrical, which is not the case in the experiments. Therefore, the teacher forcing mechanism creates serial position recall curves that do not fit the experiments. To better investigate the teacher forcing effect, the generalization of the model on other delays is represented in Figure A.6.



(a) Delay of 5 timesteps



(b) Delay of 15 timesteps

Figure A.6: Generalization of the model on other delays, for a BRC model trained with teacher forcing.

For smaller delays, the serial position recall curve has a bow-shaped curves as in experiments, but the recall performances are much too small. However, the curve for longer delay has a shape that does not reflect at all what is observed in reality. In addition, recall performance is higher for long delay times than for short ones. This is not realistic at all. Indeed, for longer delays, the participants have more time to forget the items of the sequence. As remembering the items is more difficult, the performances should be lower or at most the same as in shorter delays. All these experiments highlight the fact that using teacher forcing leads to predictions further away from those intended and thus should not be considered at all in the model.

Appendix B

About the Usage of AI

This appendix discusses the use of AI in this master thesis.

Only DeepL were used solely to translate and rephrase some parts of this manuscript, but has never been used to generate text.

For coding, only chagpt has only been used to make certain figures more attractive.

Bibliography

Ahnke, J. C. (1965). Primacy and recency effects in serial-position curves of immediate recall. *Journal of Experimental Psychology*, *70*(1), 130–132. https://doi.org/10.1037/h0022013

Alcedo, J., & Prahlad, V. (2020). Neuromodulators: An essential part of survival. *Journal of Neurogenetics*, *34*(3-4), 475–481. https://doi.org/10.1080/01677063.2020.1839066

Baddeley, A. D. (1964). Immediate memory and the "perception" of letter sequences. *Quarterly Journal of Experimental Psychology*, *16*(4), 364–367. https://doi.org/10.1080/17470216408416391

Baddeley, A. D. (1968). How does acoustic similarity influence short-term memory? [Published August 1968]. *The Quarterly Journal of Experimental Psychology. A, Human experimental psychology*, *20*(3), 249–264. https://doi.org/10.1080/14640746808400159

Baddeley, A. D. (2000). The episodic buffer: A new component of working memory? *Trends in Cognitive Sciences*, *4*(11), 417–423. https://doi.org/10.1016/S1364-6613(00)01538-2

Baddeley, A. D., & Hitch, G. (1974). Working memory. In G. A. Bower (Ed.), *Recent advances in learning and motivation* (pp. 47–90, Vol. 8). Academic Press. https://doi.org/10.1016/S0079-7421(08)60452-1

Barone, P., & Joseph, J. P. (1989). Prefrontal cortex and spatial sequencing in macaque monkey. *Experimental Brain Research*, *78*(3), 447–464. https://doi.org/10.1007/BF00230234

Bayliss, D. M., Jarrold, C., Gunn, D. M., & Baddeley, A. D. (2003). The complexities of complex span: Explaining individual differences in working memory in children and adults. *Journal of Experimental Psychology: General*, *132*(1), 71–92. https://doi.org/10.1037/0096-3445.132.1.71

Beiser, D. G., & Houk, J. C. (1998). Model of cortical-basal ganglionic processing: Encoding the serial order of sensory events. *Journal of Neurophysiology*, *79*(6), 3168–3188. https://doi.org/10.1152/jn.1998.79.6.3168

Blankenship, A. B. (1938). Memory span: A review of the literature. *Psychological Bulletin*, *35*(1), 1–25. https://doi.org/10.1037/h0061086

Blumstein, D. T. (2016). Habituation and sensitization: New thoughts about old ideas. *Animal Behaviour*, *120*, 255–262. https://doi.org/10.1016/j.anbehav.2016.05.012

Botvinick, M. M., & Plaut, D. C. (2006). Short-term memory for serial order: A recurrent neural network model. *Psychological Review*, *113*(2), 201–233. https://doi.org/10.1037/0033-295X.113.2.201

Bouton, M. E., & Moody, E. W. (2004). Memory processes in classical conditioning. *Neuroscience & Biobehavioral Reviews*, *28*(7), 663–674. https://doi.org/10.1016/j.neubiorev.2004.09.001

Bramham, C. R., & Messaoudi, E. (2005). Bdnf function in adult synaptic plasticity: The synaptic consolidation hypothesis. *Progress in Neurobiology*, *76*(2), 99–125. https://doi.org/10.1016/j.pneurobio.2005.06.003

Burgess, N., & Hitch, G. J. (1999). Memory for serial order: A network model of the phonological loop and its timing. *Psychological Review*, *106*(3), 551–581. https://doi.org/10.1037/0033-295X.106.3.551

Chai, W. J., Hamid, A. I. A., & Abdullah, J. M. (2018). Working memory from the psychological and neurosciences perspectives: A review. *Frontiers in Psychology*, *9*, 401. https://doi.org/10.3389/fpsyg.2018.00401

Colom, R., Rebollo, I., Abad, F. J., & Shih, P.-F. (2006). Complex span tasks, simple span tasks, and cognitive abilities: A reanalysis of key studies. *Memory & Cognition*, *34*(1), 158–171. https://doi.org/10.3758/BF03193395

Cowan, N. (2008). What are the differences between long-term, short-term, and working memory? [Chapter 20]. In N. K. Logothetis & I. R. Paper (Eds.), *Progress in brain research* (pp. 323–338, Vol. 169). Elsevier. https://doi.org/10.1016/S0079-6123(07)00020-9

Cowan, N., Chen, Z., & Rouder, J. N. (2004). Constant capacity in an immediate serial-recall task: A logical sequel to miller (1956). *Psychonomic Bulletin & Review*, *11*(1), 120–125. https://doi.org/10.1111/j.0956-7976.2004.00732.x

Cunningham, P., Cord, M., & Delany, S. J. (2008). Supervised learning. In M. Cord & P. Cunningham (Eds.), *Machine learning techniques for multimedia* (pp. 21–49). Springer. https://doi.org/10.1007/978-3-540-75171-7_2

Dempster, F. N. (1981). Memory span: Sources of individual and developmental differences. *Psychological Bulletin*, *89*(1), 63–100. https://doi.org/10.1037/0033-2909.89.1.63

Elman, J. L. (1990). Finding structure in time [The seminal paper introducing Simple Recurrent Networks (SRNs)]. *Cognitive Science*, *14*(2), 179–211. https://doi.org/10.1207/s15516709cog1402_1

Farrell, S., Oberauer, K., Greaves, M., Pasiecznik, K., Lewandowsky, S., & Jarrold, C. (2016). A test of interference versus decay in working memory: Varying distraction within lists in a complex span task. *Journal of Memory and Language*, *90*, 66–87. https://doi.org/10.1016/j.jml.2016.03.010

Franke, M., & Degen, J. (2023, September 28). *The softmax function: Properties, motivation, and interpretation*. PsyArXiv. https://doi.org/10.31234/osf.io/vsw47

Frankish, C. R. (1995). Intonation and auditory grouping in immediate serial recall. *Applied Cognitive Psychology*, *9*(1), 5–22. https://doi.org/10.1002/acp.2350090103

Frankland, P. W., & Bontempi, B. (2005). The organization of recent and remote memories. *Nature Reviews Neuroscience*, *6*(2), 119–130. https://doi.org/10.1038/nrn1607

Geeter, F. D., Ernst, D., & Drion, G. (2024). Spike-based computation using classical recurrent neural networks. *Neuromorphic Computing and Engineering*, *4*(2), 024007. https://doi.org/10.1088/2634-4386/ad473b

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256.

Goldman-Rakic, P. S. (1995). Cellular basis of working memory. *Neuron*, *14*(3), 477–485. https://doi.org/10.1016/0896-6273(95)90304-6

Greene, R. L. (1989). Immediate serial recall of mixed-modality lists. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *15*(2), 266–274. https://doi.org/10.1037/0278-7393.15.2.266

Greene, R. L. (1991). The ranschburg effect: The role of guessing strategies. *Memory & Cognition*, *19*(3), 313–317. https://doi.org/10.3758/BF03211155

Guilford, J. P., & Dallenbach, K. M. (1925). The determination of memory span by the method of constant stimuli. *The American Journal of Psychology*, *36*, 621–628. https://doi.org/10.2307/1413916

Hartley, T., Hurlstone, M. J., & Hitch, G. J. (2016). Effects of rhythm on memory for spoken sequences: A model and tests of its stimulus-driven mechanism. *Cognitive Psychology*, *87*, 135–166. https://doi.org/10.1016/j.cogpsych.2016.05.001

Hashemi, M. (2019). Enlarging smaller images before inputting into convolutional neural network: Zero-padding vs. interpolation. *Journal of Big Data*, *6*, 98. https://doi.org/10.1186/s40537-019-0263-7

Henson, R. N. A. (1996). *Short-term memory for serial order* [Unpublished doctoral dissertation, MRC Applied Psychology Unit, University of Cambridge, England. Copyright 1996 by R. N. A. Henson].

Henson, R. N. A. (1998). Short-term memory for serial order: The start–end model. *Cognitive Psychology*, *36*(2), 73–137. https://doi.org/10.1006/cogp.1998.0685

Henson, R. N. A., Norris, D., Page, M., & Baddeley, A. D. (1996). Unchained memory: Error patterns rule out chaining models of immediate serial recall. *Quarterly Journal of Experimental Psychology: Section A*, *49*(1), 80–115. https://doi.org/10.1080/713755612

Hertzog, C., McGuire, C. L., & Lineweaver, T. T. (2010). Aging, attributions, perceived control, and strategy use in a free recall task. *Aging, Neuropsychology, and Cognition*, *5*(2), 85–106. https://doi.org/10.1076/anec.5.2.85.601

Hitch, G. J. (2010). Temporal grouping effects in immediate recall: A working memory analysis. *The Quarterly Journal of Experimental Psychology*, *63*(1), 116–139. https://doi.org/10.1080/713755609

Hitch, G. J., Burgess, N., Towse, J. N., & Culpin, V. (1996). Temporal grouping effects in immediate recall: A working memory analysis. *Quarterly Journal of Experimental Psychology: Section A*, *49*(1), 140–158. https://doi.org/10.1080/713755611

Hoareau, V., Portrat, S., Oberauer, K., Lemaire, B., Plancher, G., & Lewandowsky, S. (2017). Computational and behavioral investigations of the sob-cs removal mechanism in working memory. *Journal of Memory and Language*, *96*, 19–36. https://doi.org/10.1016/j.jml.2017.05.005

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions [Seminal paper on the vanishing gradient problem in RNNs, Cited by: 2009 (Crossref)]. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*(2), 107–116. https://doi.org/10.1142/S0218488598000094

Hurlstone, M. J. (2020, July). *Serial recall* [Available at OSF Preprints]. https://doi.org/10.31219/osf.io/4w8fu

Jain, L. C., & Medsker, L. R. (Eds.). (1999, December). *Recurrent neural networks: Design and applications* [Original publication date: December 20, 1999]. CRC Press.

Kantowitz, B. H., Ornstein, P. A., & Schwartz, M. (1972). Encoding and immediate serial recall of consonant strings. *Journal of Experimental Psychology*, *93*(1), 105–110. https://doi.org/10.1037/h0032500

Kay, H. (2001). Learning of a serial task by different age groups. *Quarterly Journal of Experimental Psychology Section A*, *54*(4), 1085–1100. https://doi.org/10.1080/17470215108416792

Klein, K. A., Addis, K. M., & Kahana, M. J. (2005). A comparative analysis of serial and free recall. *Memory & Cognition*, *33*(5), 833–839. https://doi.org/10.3758/BF03193078

Kowialiewski, B., & Majerus, S. (2024). Free time, sharper mind: A computational dive into working memory improvement. *Cognitive Psychology*, *150*, 101701. https://doi.org/10.1016/j.cogpsych.2024.101701

Kreutzer, J. S., DeLuca, J., & Caplan, B. (2011). Serial recall. In *Encyclopedia of clinical neuropsychology* (pp. 2265–2266). Springer.

Lee, N. (2004). The neurobiology of procedural memory. In J. L. M. Jr. & R. P. Kesner (Eds.), *The neurobiology of learning* (1st ed.). Routledge.

Logie, R. H., Camos, V., & Cowan, N. (2021). *Working memory: State of the science*. Oxford University Press.

McCormack, T., Brown, G. D. A., & Vousden, J. I. (2000). Children's serial recall errors: Implications for theories of short-term memory development [Starting page: 237]. *Journal of Experimental Child Psychology*, *76*(3), 237–253. https://doi.org/10.1006/jecp.1999.2549

Mewhort, D. J. K., Popham, D., & James, G. (1994). On serial recall: A critique of chaining in the theory of distributed associative memory. *Psychological Review*, *101*(3), 534–538. https://doi.org/10.1037/0033-295X.101.3.534

Miller, G. A. (1994). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, *101*(2), 343–352. https://doi.org/10.1037/0033-295X.101.2.343

Milner, P. (2003). A brief history of the hebbian learning rule. *Canadian Psychology / Psychologie canadienne*, *44*(1), 5–9. https://doi.org/10.1037/h0085817

Murdock, J., B. B. (1987). Serial-order effects in a distributed-memory model. In D. S. Gorfein & R. R. Hoffman (Eds.), *Memory and learning: The ebbinghaus centennial conference* (pp. 277–310). Lawrence Erlbaum Associates, Inc.

Nadim, F., & Bucher, D. (2014). Neuromodulation of neurons and synapses. *Current Opinion in Neurobiology*, *29C*, 48–56. https://doi.org/10.1016/j.conb.2014.05.003

Oberauer, K., & Lewandowsky, S. (2014). Further evidence against decay in working memory. *Journal of Memory and Language*, *73*, 15–30. https://doi.org/10.1016/j.jml.2014.02.003

Oberauer, K., Lewandowsky, S., Farrell, S., Jarrold, C., & Greaves, M. (2012). Modeling working memory: An interference model of complex span. *Psychonomic Bulletin & Review*, *19*(5), 779–819. https://doi.org/10.3758/s13423-012-0272-4

Page, M. P. A., & Norris, D. (1998). The primacy model: A new model of immediate serial recall. *Psychological Review*, *105*(4), 761–781.

Ratcliff, R., & McKoon, G. (1988). A retrieval theory of priming in memory. *Psychological Review*, *95*(3), 385–408. https://doi.org/10.1037/0033-295X.95.3.385

Rice, G. A., & Robinson, D. O. (1975). The role of bigram frequency in the perception of words and nonwords. *Memory & Cognition*, *3*(5), 513–518. https://doi.org/10.3758/BF03197523

Ricker, T. J., Sandry, J., Vergauwe, E., & Cowan, N. (2020). Do familiar memory items decay? *Journal of Cognition*, *3*(1), 1–15.

Ricker, T. J., Vergauwe, E., & Cowan, N. (2016). Decay theory of immediate memory: From brown (1958) to today (2014). *The Quarterly Journal of Experimental Psychology*, *69*(10), 1969–1995. https://doi.org/10.1080/17470218.2014.914546

Rosen, V. M., & Engle, R. W. (1997). Forward and backward serial recall. *Intelligence*, *25*(1), 37–57. https://doi.org/10.1016/S0160-2896(97)90006-4

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Salem, F. M. (2022). Gated RNN: The gated recurrent unit (GRU) RNN. In *Recurrent neural networks* (pp. 85–110). Springer. https://doi.org/10.1007/978-3-030-89929-5_5

Saumier, D., & Chertkow, H. (2002). Semantic memory. *Current Neurology and Neuroscience Reports*, *2*(6), 516–522. https://doi.org/10.1007/s11910-002-0039-9

Seger, C. (2018). *An investigation of categorical variable encoding techniques in machine learning: Binary versus one-hot and feature hashing* [Master's thesis]. KTH Royal Institute of Technology [School of Electrical Engineering and Computer Science (EECS)].

Shlens, J. (2014, April 8). *Notes on Kullback-Leibler divergence and likelihood*. arXiv: 1404.2000 [cs.IT]. https://doi.org/10.48550/arXiv.1404.2000

Squire, L. R. (1992). Declarative and nondeclarative memory: Multiple brain systems supporting learning and memory [Crossmark: Check for Updates]. *Journal of Cognitive Neuroscience*, *4*(3), 232–243. https://doi.org/10.1162/jocn.1992.4.3.232

Sun, R. (Ed.). (2023). *The cambridge handbook of computational cognitive sciences* (2nd ed.) [Online publication date: April 2023]. Cambridge University Press. https://doi.org/10.1017/9781108755610

Tulving, E. (2002). Episodic memory: From mind to brain. *Annual Review of Psychology*, *53*, 1–25. https://doi.org/10.1146/annurev.psych.53.100901.135114

Vecoven, N., Ernst, D., & Drion, G. (2021). A bio-inspired bistable recurrent cell allows for long-lasting memory. *PLOS ONE*, *16*(6), e0252676. https://doi.org/10.1371/journal.pone.0252676

Walen, S. R. (1970). Recall in children and adults [Starting page: 94]. *Journal of Verbal Learning and Verbal Behavior*, *9*(1), 94–100.

Wang, S.-C. (2003). Artificial neural network. In *Interdisciplinary computing in java programming* (pp. 81–100, Vol. 743). Springer. https://doi.org/10.1007/978-1-4615-0377-4_5

Waugh, N. C. (1961). Free versus serial recall. *Journal of Experimental Psychology*, *62*(5), 496–502. https://doi.org/10.1037/h0043891

Werbos, P. J. (1990). Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550–1560. https://doi.org/10.1109/5.58337

Wingfield, A., & Byrnes, D. L. (1981). *The psychology of human memory* (1st ed.) [444 pages]. Academic Press.

Wu, Y., & Feng, J. (2018). Development and application of artificial neural network [Published online: 30 December 2017]. *Wireless Personal Communications*, *102*(2), 1645–1656. https://doi.org/10.1007/s11277-017-5224-x

Zlotnik, G., & Vansintjan, A. (2019). Memory: An extended definition. *Frontiers in Psychology*, *10*, 2523. https://doi.org/10.3389/fpsyg.2019.02523