A Recurrent Memory Model Implementation of Reinforcement Learning

Kinsey Antonina Church

A thesis submitted to the University of Ottawa

in partial fulfillment of the requirements for the

Degree of Master of Arts in Experimental Psychology

School of Psychology

Faculty of Social Sciences

University of Ottawa

Abstract

Reinforcement Learning (RL) provides a robust framework for understanding how humans and animals learn from and adapt to our environments through trial and error. It is comprised of two processes that work together, exploration and exploitation. This thesis presents an implementation of these two aspects of RL using recurrent neural networks. Several different challenges were approached including one-to-many problems, nonlinearly separable problems, and the generation and representation of random behaviour. The first study is an implementation of exploitation that is capable of cycling through previously learned behaviours and stabilizing on the correct one for each given problem. This is achieved with contextual tags and a unit that represents environmental feedback. It is able to solve nonlinearly separable and one-to-many problems. The second study is an implementation of exploration that is able to randomly select from available options and adapt based on the feedback from those decisions. It tackles the question of how to generate and represent random behaviour, as well as how to represent and apply reward. The implementations and techniques detailed in this thesis could be applied to many other similar models of cognitive behaviour. Further research could experiment by combining these two models and investigating the implementation of different trade-off strategies.

Acknowledgements

First, a huge thank you to everyone I have met during grad school, from my cohort and my professors to my co-authors. I have learned so much from all of you over the past five years and will always remember grad school fondly because of you.

To the INSPIRE RAs, thank you for the camaraderie and support. I will miss this team so much!

To my family and friends, I would not have finished without you. Thank you to my amazing parents Patti and Steve and my bros Des and Nikita for your endless support. To my best friends Jamey and Addie, thank you for always being there when I needed to rant. To my amazing partner Kris, thank you for keeping me going every day. I love you all, here's to the next chapter.

Finally, a huge thanks to all of the members of the CONEC Lab. Thank you to my main co-author over the years Matt for the countless coding, brainstorming, and writing sessions. Thank you to Damiem for being so encouraging when I was learning Python and helping me write my very first abstract (for SQRP). Thank you to Marija for being a lovely co-author and making running the randomness simulations fun. Finally, thank you to Sylvain for being such a supportive supervisor and mentor. I deeply appreciate everything you have done for me and the lab over the years, and I will never forget my time at the CONEC Lab.

Contribution of Authors

The studies of which this dissertation is comprised have been prepared as two manuscripts, which have both been subject to peer-review:

Study 1 (Chapter 2)

Church, K.A., Ross, M., & Chartier, S. (2020, July). Using a Bidirectional Associative Memory and Feature Extraction to model Nonlinear Exploitation Problems. In Steward, T. C. (Ed.). *Proceedings of the 19th International Conference on Cognitive Modelling* (pp. 37-42). University Park, PA: Applied Cognitive Science Lab, Penn State.

Study 2 (Chapter 3)

Church, K., Bolic, M., & Chartier, S. (2024, July). Generating Distributed Randomness using Artificial Neural Networks. In *Proceedings of the 46th Annual Conference of the Cognitive Science Society: CogSci 2024* (pp. 4306-4313). Cognitive Science Society.

As the first author, I was responsible for the exploration and testing of the neural network models, the data collection, and the report writing. For Study 1, my second author Dr. Matthew Ross, graduate student in my lab, helped me with coding and troubleshooting. The senior author, Dr. Sylvain Chartier provided the code for the original neural network models, as well as support and guidance for my study design and assistance editing. For Study 2, as first author, I programmed the input stimuli, changed the network from bipolar to binary, and ran most of the simulations and data collection. My second author Marija Bolic, honours student in my lab, helped me run simulations and collect data. The senior author, Dr. Sylvain Chartier, provided the base code for the neural network models, helped with running simulations to verify results, and provided guidance and support for the study design and execution.

List of Figures

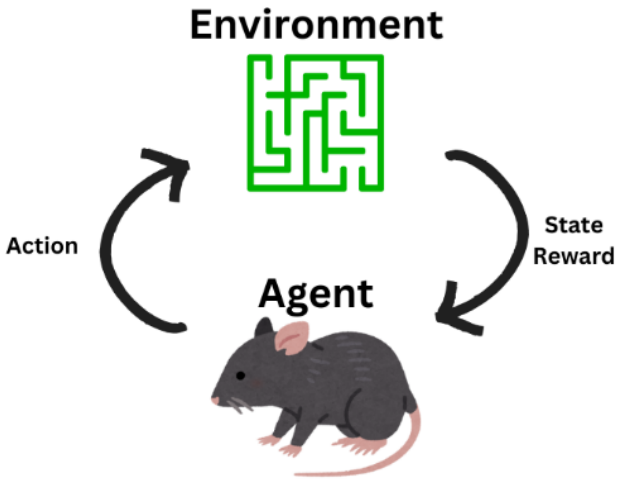
Table of Contents

CHAPTER 1: Introduction

Reinforcement Learning (RL) is a concept at the intersection of cognitive psychology and machine learning. It provides a robust framework for understanding how agents learn from and adapt to their environments through trial and error (Sutton & Barto, 1999; Sutton & Barto, 2017). Unlike supervised learning, which relies on labeled data to train models, and unsupervised learning, which seeks to uncover hidden patterns in unlabeled data, RL falls somewhere in the middle. It involves an agent learning to make decisions by receiving feedback in the form of rewards or penalties. It is commonly regarded as the most natural of the three, as this process mirrors the natural learning mechanisms observed in humans and animals, where behaviours are shaped and optimized based on feedback from the environment (Akanksha et al., 2021). For example, a rat (agent) in a maze (environment) will receive feedback in the form of reward when it completes the maze. It must act based on its current position (state) in the maze and learn to maximize the expected reward (see Fig. 1).

Figure 1

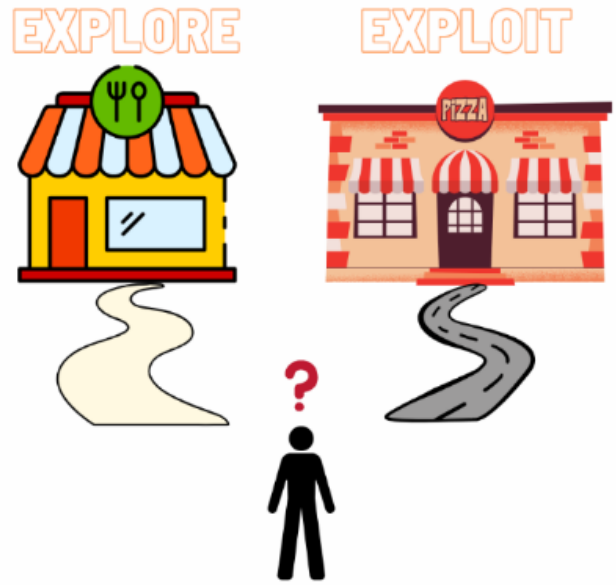
A diagram of Reinforcement Learning with a rat learning a maze



At the heart of RL is a trade-off known as the Exploration-Exploitation (E-E) dilemma; where a human or an animal must find a balance between exploring their environment in search of new options and exploiting their current knowledge in order to solve a task (Kaelbling, Littman & Moore, 1996; see Fig. 2). The risk is that when exploiting, one may be missing out on a better option and when exploring, one might be wasting time or effort when the best option is already known (Hills et al., 2015). As humans, we make these kinds of decisions every day without thinking twice, but this is one of the interesting cognitive trade-offs responsible for our survival and evolutionary progress (Del Giudice & Crespi, 2018).

Figure 2

Visualization of the Exploration-Exploitation dilemma where one must choose between an old favourite restaurant and a new, riskier option



E-E is important when studying how we learn from, and interact with, our environment. It has been studied from many different perspectives, including neuroscience (Laureiro-Martínez et al., 2014), psychology (Stafford et al., 2012), and ecology (Monk et al., 2018). Many researchers believe that the E-E trade-off exists on a spectrum from pure exploration to pure exploitation, and every level in between (Mehlhorn et al., 2015). Previous work in neuroimaging showed that the two processes could not be fully disentangled (Blanchard & Gershman, 2018). We can therefore think of exploration and exploitation as two processes that are closely related rather than opposites.

Many formal models have been proposed that model E-E in RL throughout the years such as multi-armed bandits (Audibert et al., 2009; Besbes, Gur & Zeevi, 2019; Brown et al., 2022; Vakili, Liu & Zhao, 2013) and Thompson sampling (Chapelle & Li, 2011; Ferreira, Simchi-Levi & Wang, 2018), but the most popular is a branch of RL known as Q-Learning. Q-Learning is a good model for learning optimal policies and is commonly used as a benchmark for the E-E dilemma. It consists of an algorithm that learns by adapting and does not require prior knowledge of its environment (Watikins & Dayan, 1992). In Q-Learning, when the agent is faced with a decision, they have the option of exploiting the optimal action based on current knowledge or exploring by choosing a lesser-known action. It dynamically adjusts the balance between exploiting well-known, highly rewarded options and randomly exploring some of the lesser-known options (Watkins, 1989). This is analogous to how humans and animals learn from our actions, decisions, and environment throughout the course of our lives.

More precisely, Q-Learning starts by initializing a Q-table at zero, which is where reward is stored as the agent receives it. To learn, the agent must observe its current state to select an action. The action is selected by a policy known as ϵ-greedy, where a random number is

generated and if the number is below a set threshold (ϵ), the action is selected randomly (exploration). If the number is above that threshold, the action with the highest value for the current state in the Q-table is chosen (most rewarded known action; exploitation). From this action, it receives a numerical value reward (positive or negative) for that chosen action and updates the Q-table for the specific state-action pair that was just chosen. Then, the agent transitions to the new state as a result of that action. These steps are repeated until a certain number of trials have been surpassed or until the learning converges (Watkins, 1989).

The aim of this thesis is to create a neural implementation of Q-Learning, that is able to complete the same tasks. To create a neural implementation of Q-Learning, we need networks that can perform the same actions described above. For example, to exploit the state-action rewards that have already been learned, the network will need to be able to output a list of actions for a given task. As the environment changes, it should be able to change its behaviour accordingly, meaning it will be able to stabilize on the correct response for a given problem.

Another challenge is that the network must be able to randomly select from behaviours when it is choosing an action (exploring). For the proposed project, exploration will be defined as randomly deciding between all available options, as seen in Q-Learning when the agent must choose a random action from the options available in its current state. Research in neuroscience and cognition has shown that humans also generate behaviour randomly when approaching a novel stimulus or difficult task (Daw et al., 2006; Tervo et al., 2014).

Historically, both internal and external randomness in neural network models are represented by a single number generated by a computer (for example, in Hélie & Sun (2010)). Surprisingly, there are few studies proposing a neural architecture that implements randomness using network-wide representation created by the internal dynamics of the network itself. This

distributed representation is preferable to relying on an external random number generator because it is closer to how we store information in the brain, across multiple neurons instead of stored in a single one (Chen, Nelson & Hsu, 2015; O'Toole et al., 2005; Rissman & Wagner, 2012; Small et al., 1995). We will be addressing how to generate this distributed randomness internally using the dynamic properties of the network.

Finally, once learning has taken place, decisions are then based on the choice that has proven most likely to be rewarded (Héricé et al., 2016). Therefore, the network should be able to bias the choices based on feedback from the environment, as is done in Q-Learning by the policy updating over time. The final implementations should be able to exploit time series of previously learned responses (similar to how the Q-table works) and explore by generating randomness (similar to exploration using the ϵ-greedy policy) that can be biased through feedback from reward. The present thesis investigates both implementations, which can then be combined in future studies to model the full E-E dilemma or replicate Q-Learning.

Research Objectives

RL and more specifically, Q-Learning, are widely used in machine learning and other fields, but few models exist in psychology. This project will create a dynamical neural implementation inspired by Q-Learning, with added properties that make it more relevant to cognition. One added property is the use of internal randomness to drive exploration. Another new property is the use of context when exploiting to be able to solve tasks, while being robust to high levels of variability. This thesis is divided into two studies, which will hopefully pave the way to represent the entire trade-off.

Study 1 is an implementation of how previously learned responses are exploited, similar to the Q-table being utilized in Q-Learning. The network should be able to choose from

previously learned behaviours when faced with a task and to keep trying different learned behaviours until the task is solved. The goal is to be able to solve problems with the level of variability that we find in the real world, including nonlinearly separable problems (problems that cannot be solved with a single straight line and require more complex boundaries to separate categories). For this to be implemented, many challenges must be addressed. For example, how the network will be able to give different responses when presented with fixed inputs without altering the weights or learning anything new. This is known as the one-to-many problem because the network learns many different sequences of behaviours and must switch between them without the input changing. The desired network should be able to solve different tasks and give different responses with the same base knowledge, the way that humans and animals can. Another challenge is how context about the task from the external world and other information from the environment can be incorporated.

Study 2 is an implementation of the how randomness can be generated, represented, and biased, similar to exploration in Q-Learning. The aim is to represent how actions are selected randomly in Q-Learning, in a recurrent neural network model that is compatible with the architecture from Study 1. This implementation will include how to generate randomness in order to select from available behaviours and how this randomness can be biased with feedback from the environment. This study investigates how to represent randomness in a distributed fashion, how to represent decisions made with little to no prior knowledge, and how these decisions can be biased with reward from the environment. The discussion touches on a few examples of how these implementations can be combined and applied and discusses their limitations.

CHAPTER 2: Study 1 - Using a Bidirectional Associative Memory and Feature Extraction to model Nonlinear Exploitation Problems

Each day we are faced with a decision of maximizing our resources by using our current knowledge to learn new things. Should we go to the new restaurant that just opened around the corner or stick to an old, reliable favourite? This is known as the exploration-exploitation dilemma and it is at the heart of reinforcement learning. The present study looks at the exploitation half of this problem and aims to implement it in a biologically plausible recurrent associative memory model. In the framework of Artificial Neural Networks, exploitation is observed when the network can iterate through many learned responses and stabilize on the correct one to solve a given task. This is a process akin to being able to switch from a cyclic to a point attractor. More precisely, Bidirectional Associative Memory (BAM) is used to accomplish such tasks where the context dictates which attractor the network should converge to. For simple independent tasks, the BAM is sufficient. However, for overlapping tasks, the task becomes nonlinearly separable. Therefore, the BAM needs an extra unsupervised layer to extract unique features from the inputs. These features combined with input are then sent to BAM where it can learn the different attractors adequately. This network was able to stabilize on the correct responses of tasks that involved time series of varying lengths, overlap, and levels of correlation; the variability one would expect from the real world.

Keywords: exploitation/exploration; recurrent associative memory; one-to-many associations.

Introduction

In order to increase their chances of survival, many organisms have evolved various complex mechanisms to solve problems, make decisions, and learn about the environment around them. These mechanisms come with several different trade-offs, where increasing proficiency in one area may result in a lack of ability in another area. One of these problems in cognition is the exploration-exploitation dilemma (Hills et al., 2015). This is when an organism must choose whether it wants to employ something it has already learned (exploitation) or explore its environment to learn new things (exploration). For example, when a young child is first learning how to open different doors. They may start by trying the few ways they already know to open a door (exploitation), such as pulling and pushing. If these behaviours fail, however, they will have to start looking for new solutions (exploring). Understanding exactly how animals and humans switch between these two processes and how each of them works is crucial to understanding how we adapt, learn, behave, and survive (Hills et al., 2015). This trade-off is one of many phenomena affected by Reinforcement Learning (RL).

Previous studies have proposed several methods to solving this problem from an "optimal solution" perspective. For example, Hwang, Chiou and Wu (2003) created an Artificial Neural Network (ANN; an evaluation predictor) that solves the exploration-exploitation problem that was able to both explore and exploit to find an optimal solution to their task (balancing a pendulum). They used the mean of a normal distribution to represent the exploitation function, which may be a bit too basic to capture the exploitation process. This study and many others focus too much on obtaining a desired state rather than the underlying process of how exploitation (and exploration) take place or attempt to find a biologically plausible solution (Cohen, McClure & Yu, 2007; Tilahun, 2019).

Another promising study was conducted by Lew, Rey, and Zanutto (2013). It outlines a biologically plausible model of switching between exploration and exploitation depending on changes in the environment and rewards from the dopaminergic system. Their model is a computational framework of how exploration and exploitation are selected and relates this process back to the reward system in the human brain, focusing on the selection only. Many studies have a similar focus and are more about the overall model rather than how each process works (Gershman & Niv, 2015; Hallquist & Dombrovski, 2019).

An important consideration overseen by these studies is the context in which the decision takes place. Empirical studies have shown how the environment is an important factor in predicting how an animal will behave and how it will exploit its resources (Naruse et al., 2018). Some studies have shown how the decisions humans and animals make influence their ecosystem and vice versa, from small-scale interactions among individual agents to large-scale adaptive systems, like population dynamics (Monk et al., 2018). The context in which an animal is found clearly plays a role on how it chooses to exploit and explore.

The present study looks at the exploitation half of the dilemma using environmental context as a cue. For instance, an animal faces the exploration-exploitation dilemma while foraging for food (Hills et al., 2015). Is it more efficient to explore the area for new sources of food or stick to a learned set of locations where food has been found in the past?

The most well-known example of exploitation has been in Q-Learning (Watkins & Dayan, 1992). In this implementation, the best outcome is simply the one that has the maximum possible reward; MaxQ. This works best when the reinforcement is non-binary. However, in the situation of multiple possible outcomes, the agent should be able to test different possible behaviours until the correct one has been found. For example, if the animal chooses to exploit

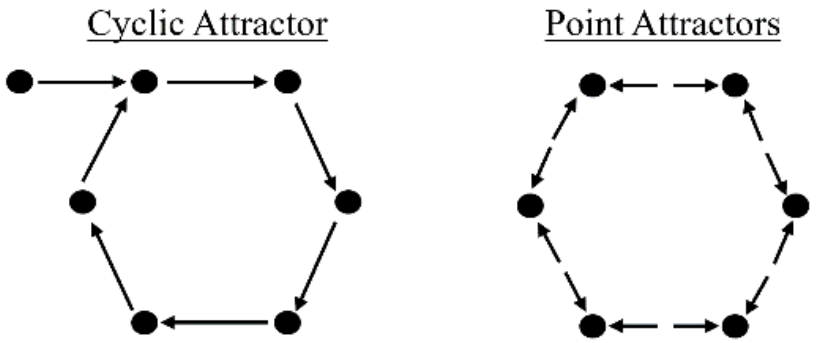
their already-known resources, they cycle through them until the problem is solved (they find food). Once it is solved, they will stop trying different behaviours and focus on the successful one.

In ANNs, the problem can be framed as how the network can generate a different output (behaviour) when the input (context) remains the same? In ANNs, if a given input is sent to the network, it will always give the same associated output. However, if that output is not satisfying, the network must give a different output even though it is still the same input that is presented. One way to solve this problem is to use a cycle attractor. This process can be modeled by the network generating various outputs from the same input to represent the different learned behaviours. For example, input 1 generates output 1, which generates output 2, all the way until output *n*. Output *n*, the final output, is then associated with the output 1, creating a loop of learned patterns. This would be like a squirrel having a series of berry bushes where it knows food can be found and going from bush to bush until it finds some.

This can be easily implemented in a Bidirectional Associative Memory (BAM) as a multistep time series (Chartier & Boukadoum, 2006). However, a problem arises when the network must stabilize on a specific output *i*. Therefore, each of the patterns (output) must also be stored as point attractors in the network. This represents a one-to-many situation where for the same output *i* the network must generate output *i*+1 (next pattern; cyclic attractor) or output *i* (same pattern; point attractor). See Figure 3 for a diagram of cyclic and point attractors.

Figure 3

Illustration of Cyclic and Point Attractors



The solution may reside in the inclusion of context that can dictate which state the network should be in and therefore give distinctive features. From the feedback given by the environment, the network should be able to switch between a cyclic (repeating series of outputs) and a point attractor (single output). However, instead of using context from the time series itself (Elman, 1990), the context has to be extracted from the reading of the environment; a process proposed in (Jordan, 1997) and successfully implemented in BAM (D. Rolon-Merette, T. Rolon-Mérette & Chartier, 2018). By having a unique context as the initial input of the time series, it can also be used to distinguish multiple cyclic attractors (different time series).

The BAM can be used to store various attractors including point, cyclic, and chaotic attractors (Chartier, Boukadoum & Amiri, 2009), making it a well-suited candidate to perform exploitation. Figure 4 shows an example of both cyclic attractors (a to c) and fixed-point attractors (d) in the model. When the BAM is giving the next output, it is recalling associations in a cycle. In this case, it recalls ‘A’ from ‘1’ and proceeds in order (‘B’, ‘C’, etc.). However, when it needs to stabilize, it switches to a fixed-point attractor. In this example, the environment

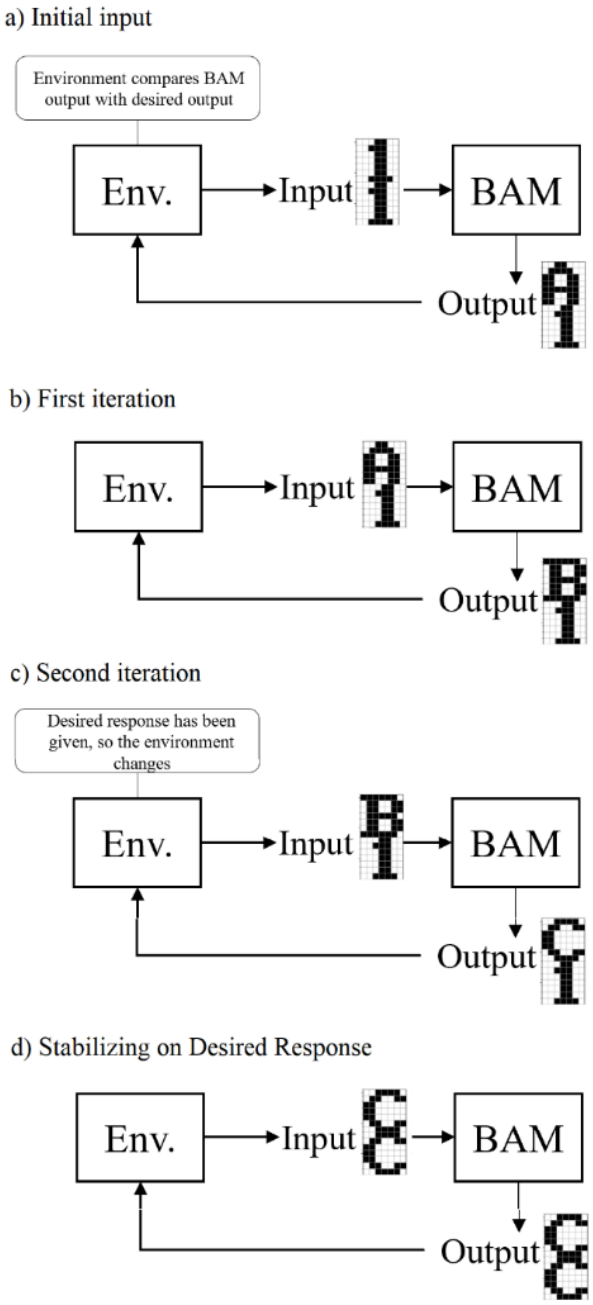
gives ‘C’ associated with itself as a response, which is different from ‘C’ in the ‘1’ context, causing the BAM to switch from a cyclic attractor to a fixed-point attractor. In order to focus exclusively on the exploitation phase, it is assumed that some level of exploration has already taken place. In other words, that series of behaviours have already been learned.

Putting Figure 4 in the context of the foraging example, ‘1’ could represent hunger, and the other patterns represent learned behaviours, such as searching for food in different locations; ‘A’, ‘B’, ‘C’, etc. When food is located, the environment lets the animal know to stop cycling through its learned behaviours (checking different locations) and make the decision to stay there.

The remainder of this chapter is divided into two separate sets of simulations. Simulation I introduces the problem of modeling exploitation and switching from iteration to stabilizing on a given response using the environmental context (one-to-many association). Simulation II solves more complex and diverse situations involving nonlinearly separable associations. An overall discussion is provided at the end.

Figure 4

Flowchart Illustrating the Switching from a Cyclic Attractor to a Fixed-point Attractor using the Environment



Simulation I- Independent Time Series

Methods

To model exploitation, time series of patterns were used to represent different possible output behaviours. Each series was preceded by a unique contextual input. The network task consisted of associating multiple time series of different lengths and various levels of overlap. Moreover, the level of correlation between patterns varies vastly to encapsulate the variability found in the real world.

Stimuli. Patterns of behaviours were represented by a series of letters and the context by a number. Each pattern was placed on a 7x7 grid where a black pixel had a value of +1 and a white pixel a value of -1. Following the procedure in (D. Rolon-Merette et al., 2018) for each pattern of the time series, the associated context pattern was appended. These contexts allow the network to differentiate between different time series and the desired type of attractor (cyclic vs. point).

The various time series are illustrated in Fig. 5, with grey boxes to highlight the repeated and overlapping patterns. Series 1 is a single time series of patterns associated with the context of number '1' and each of the patterns are also associated with themselves. It represents a basic, independent time series with no overlap and is the least complex series. Series 2 increases the difficulty by having two independent time series of various lengths. Series 3 includes three independent time series of various lengths as well. Series 4 and 5 represent situations closer to reality, where the difficulty is further increased. In both examples, a given pattern is associated not only to two settings (the cyclic and the point attractor) but to four (series 4) or three (series 5) settings. Series 4 shows a setting where patterns can belong to various solutions (overlapping),

while series 5 shows a series that is a subtype of the other. In both cases, the problem is no longer linearly solvable.

Figure 5

Stimuli for Simulations I and II

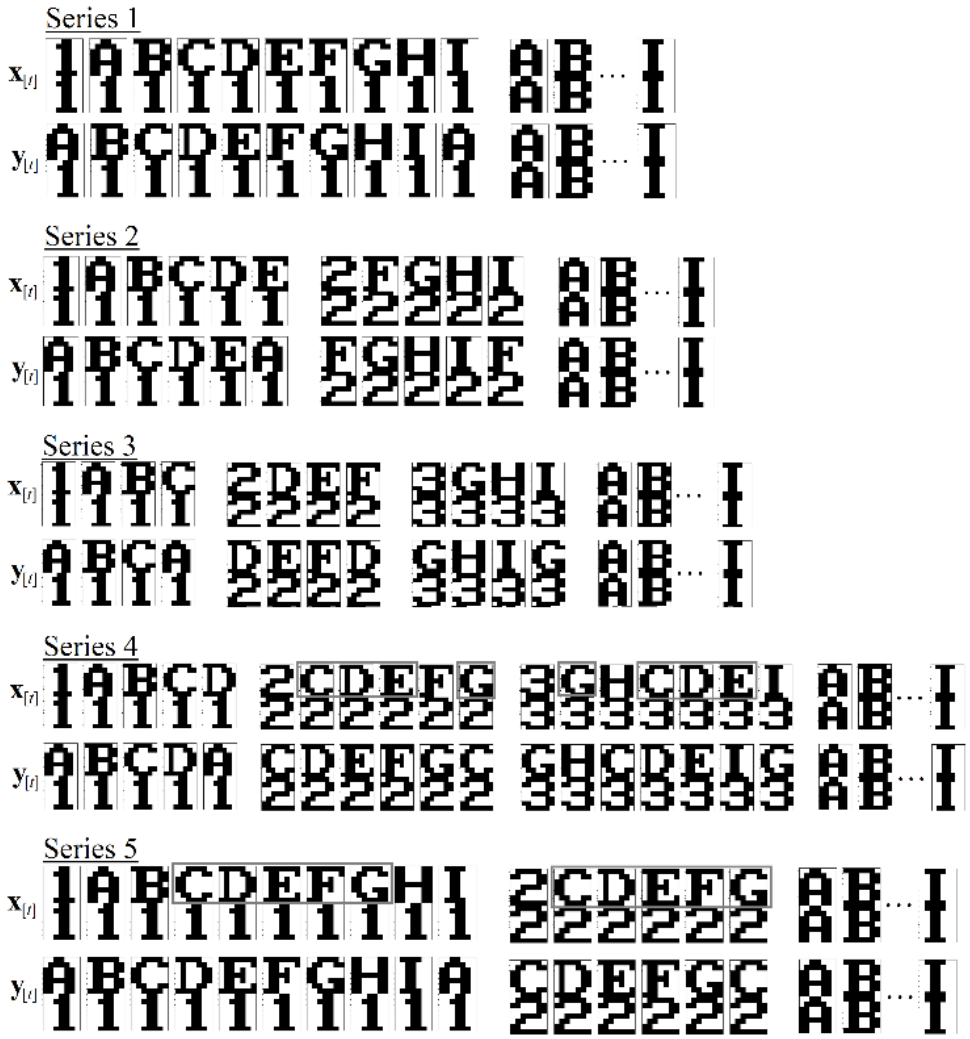
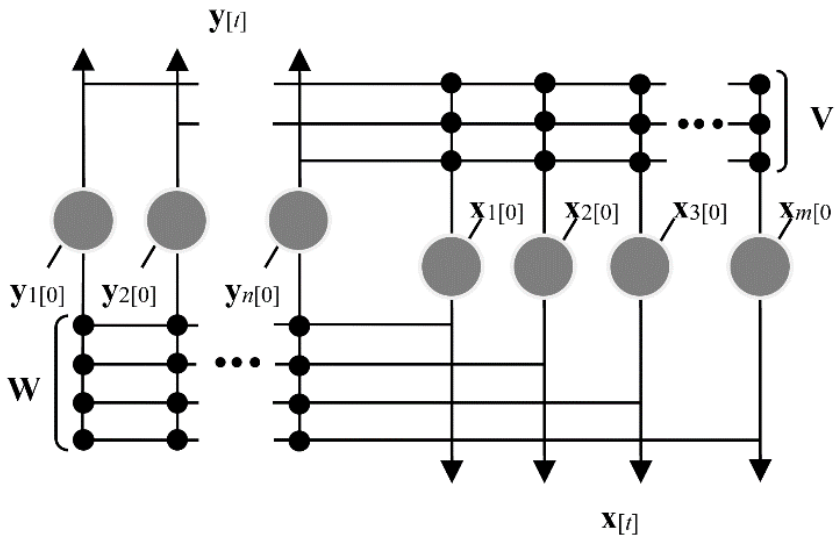


Figure 6

Architecture of the BAM



Architecture. The architecture for the modified BAM (Chartier & Boukadoum, 2006) is made of two Hopfield-like neural networks interconnected in a head-to-tail fashion to create a bidirectional flow of information (see Fig. 6). The initial vector-states are represented by $x_{[0]}$ and $y_{[0]}$, the weight matrices by W and V, the dimensionality by m and n, and t represents the time (current iteration number).

Output Function. The transmission is composed of the activation function that is then fed through the output function. The activation is obtained the usual way:

$$\mathbf{a}_{[t]} = \mathbf{W}\mathbf{x}_{[t]} \tag{1a}$$

$$\mathbf{b}_{[t]} = \mathbf{V}\mathbf{x}_{[t]} \tag{1b}$$

Once this linear activation is obtained it goes through the nonlinear output function defined as (2a) and (2b):

$$\forall i,\ldots, n,\ \mathbf{y}_{i[t+1]}=f\left(\mathbf{a}_{i[t]}\right)=\begin{cases}1, & \text{if } \mathbf{a}_{i[t]}>1\\ -1, & \text{if } \mathbf{a}_{i[t]}<-1\\ (\delta+1)\mathbf{a}_{i[t]}-\delta\mathbf{a}_{i[t]}^{3}, & \text{else}\end{cases} \tag{2a}$$

$$\forall i,\ldots, m,\ \mathbf{y}_{i[t+1]}=f\left(\mathbf{b}_{i[t]}\right)=\begin{cases}1, & \text{if } \mathbf{b}_{i[t]}>1\\ -1, & \text{if } \mathbf{b}_{i[t]}<-1\\ (\delta+1)\mathbf{b}_{i[t]}-\delta\mathbf{b}_{i[t]}^{3}, & \text{else}\end{cases} \tag{2b}$$

where n and m are the number of units in each layer of the network, i is the index unit, δ is the transmission parameter, and **a** and **b** are the activations. The output is a cubic function with saturating limits at ±1.

Learning Function. The connection weights are modified following a Hebbian/anti-Hebbian rule (Bégin & Proulx, 1996; Storkey & Valabregue, 1999) and can be represented as:

$$\mathbf{W}_{[k+1]}=\mathbf{W}_{[k]}+\eta(\mathbf{y}_{[0]}-\mathbf{y}_{[t]})(\mathbf{x}_{[0]}+\mathbf{x}_{[t]})^{\mathrm{T}} \tag{3a}$$

$$\mathbf{V}_{[k+1]}=\mathbf{V}_{[k]}+\eta(\mathbf{y}_{[0]}-\mathbf{y}_{[t]})(\mathbf{x}_{[0]}+\mathbf{x}_{[t]})^{\mathrm{T}} \tag{3b}$$

where **W** and **V** represent the two sets of weight connections, $\mathbf{x}_{[0]}$ and $\mathbf{y}_{[0]}$ the initial inputs, η the learning parameter, and k the current learning trial. The η is calculated using the following inequality function to guarantee that the learning will converge (Chartier & Boukadoum, 2006):

$$\eta<\frac{1}{2(1-2\delta)\mathrm{Max}[m,n]},\delta\neq\frac{1}{2} \tag{4}$$

Parameters. To ensure all associations are stable, delta (δ) was set to 0.2, number of iterations (t) to 1 and learning rate (η) was set to 0.008 (equation 4). The minimum Mean

Squared Error (MSE) was 10^{-8} and the maximum amount of learning trials was set to 1000 (in case of non-convergence). The weights **W** and **V** were initialized at 0.

Learning Procedure.

1. Selection of a series (Fig. 5).

2. Selection of an associated pair as the initial input for $\mathbf{x}_{[0]}$ and $\mathbf{y}_{[0]}$.

3. Computation of the outputs $\mathbf{x}_{[1]}$ and $\mathbf{y}_{[1]}$ using equations 1 and 2.

4. Weights update (**W** and **V**) according to equation 3.

5. This was repeated with each of the subsequent patterns until the MSE for all associations reached 10^{-8} or the number of trials reached 1000.

Exploitation Procedure.

1. Selection of a "desired response" for the network to stabilize on for a given time series ("initial context"). This is the response stored in the environment for comparison.

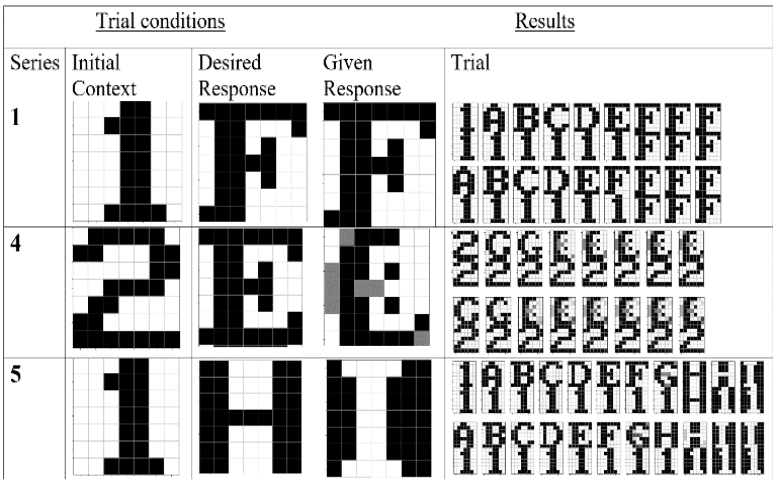
2. First pattern in the time series is used as the input for BAM ($\mathbf{x}_{[0]}$).

3. Environment compares output ($\mathbf{y}_{[t]}$) with the "desired response" for each iteration. If it does not match, the network uses this pattern as the new input ($\mathbf{x}_{[t]}=\mathbf{y}_{[t]}$) and generates the next one ($\mathbf{y}_{[t+1]}$). If it matches (BAM has found the correct pattern), the environment sends only this pattern back to the BAM. This can be seen as the environment changing when the solution to a task is found (for example, when you are trying to open a door and it opens vs. remaining closed), providing enough feedback to stabilize on this correct response.

4. The BAM recalls the pattern associated with itself and stabilizes on it instead of continuing to iterate through the series.

Figure 7

Results of Example Trials from Simulation I



Results

The BAM was able to learn and recall the first three series. It was also able to stabilize its output on any desired pattern with perfect accuracy. For example, in Figure 7 we see one trial in series 1, where ‘F’ was the correct response in time series ‘1’. The initial input was ‘1’ and it was given to the BAM, which recalled ‘A’. This output was compared to the desired response in the environment, and since it was not a match, the output became the new input and was sent back to the BAM. This continued until the correct response ‘F’ was given, where the environment changed because the task had been solved and sent ‘F’ both as input and context, to the BAM.

The network was then able to give the same output, therefore causing it to stabilize on the correct response. However, when the BAM was faced with more complex tasks (overlaps or subtype), it could no longer learn the associations properly and stabilize on the desired response.

For example, when encountering a nonlinear task as seen in series 4 and 5, the results are noisy, and the BAM often fails to learn all the time series properly. It also cannot stabilize on the correct response. This is due to the nature of the task that necessitates a nonlinear classification, a capacity lacking in a single layer BAM (Chartier & Boukadoum, 2006). Therefore, the next simulation solves this problem by adding an unsupervised layer to the BAM (D. Rolon-Mérette et al., 2018).

Simulation II- Overlapping Time Series

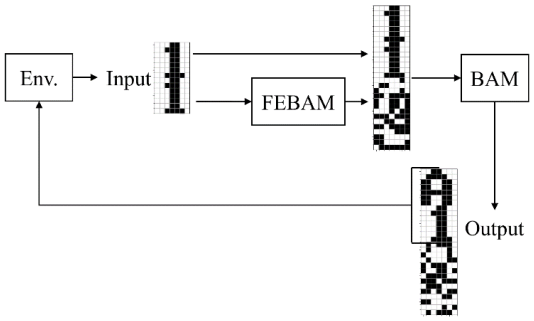
Methods

As indicated by the results from Simulation 1, a BAM alone using contexts is not sufficient to solve problems of higher complexity, meaning it cannot solve real-world problems.

Previous works have shown that a Feature Extraction Bidirectional Associative Memory (FEBAM) can be used in combination with the BAM in order to solve nonlinearly separable tasks (Tremblay et al., 2013). Therefore, the architecture of the network has to be modified accordingly. The new flowchart is shown in Figure 8.

Figure 8

Flowchart for Simulation II, Showing Unique Representation from FEBAM being Appended



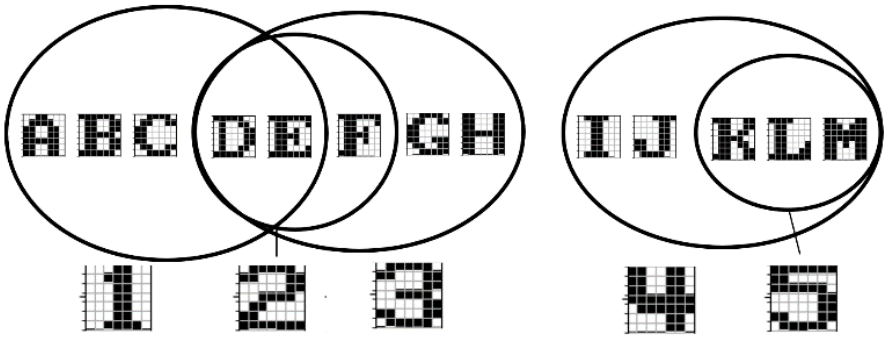
Stimuli. All series from Figure 5 were used with the addition of one additional complex series (Fig. 9). This new series includes a mix of all possible combinations of multiple, overlapping and subtype time series; representing the kind of variability found in nature. It encompasses all possible combinations and arrangements of stimuli. Figure 10 illustrates these combinations, using a Venn diagram for clarity.

Figure 9

Added Stimuli for Simulation II

Figure 10

Venn Diagram Illustrating which Rime Series are Independent, Subtype, or Overlap in Series 6



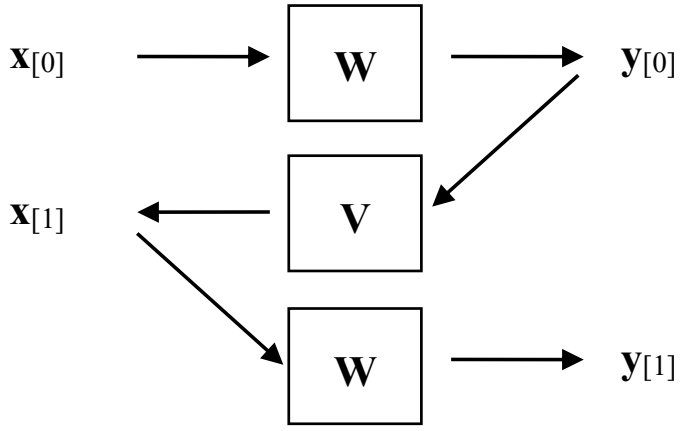
Architecture. The FEBAM is the unsupervised version of the BAM. The only difference between the two models is the absence of a set of external connections; the $\mathbf{y}_{[0]}$ inputs (see Fig. 6). This means there is only one explicit connection, $\mathbf{x}_{[0]}$, making the learning unsupervised.

Output Function and Learning Function. The transmission is obtained the same way as the BAM. Regarding the learning, because the $\mathbf{y}_{[0]}$ connections are no longer available, they are obtained by iterating once through the network as illustrated in Fig. 11. The weights of the FEBAM are then updated exactly as the BAM (equations 3a, 3b).

Parameters. For both BAM and FEBAM, the transmission parameter (δ) was set to 0.2 to ensure the fixed points would be stable. The learning parameter (η) was set to 0.004. The maximum amount of learning trials was set to 1000, and the minimum required MSE was 10^{-8}. The weights of the FEBAM were randomly initialized between -2 and 2 (Tremblay et al., 2013).

Figure 11

Output Iterative Process used for the FEBAM Learning



Learning Procedure.

1. Selection of a series (Fig. 4 and 8).

2. Selection of an associated pair as the initial input for the FEBAM.

3. The output of the FEBAM is then appended to the initial input and sent to the BAM (Fig. 8).

4. The BAM outputs the next associated patterns.

5. Weights update for both the FEBAM and BAM.

6. This was repeated with each of the subsequent patterns until the MSE for all associations reached 10^{-8} or the number of trials reached 1000.

Exploitation Procedure.

1. Selection of a "desired response" for the network to stabilize on for a given time series ("initial context"). This is the response stored in the environment for comparison.

2. First pattern in the time series is an input for FEBAM.

3. The output of the FEBAM is then appended to the initial pattern and sent to the BAM.

4. Environment compares output of the BAM ($\mathbf{y}[t]$) with "desired response" for each iteration t. If it does not match, the network uses this pattern as the new input ($\mathbf{x}[t]=\mathbf{y}[t]$) and generates the next one ($\mathbf{y}[t+1]$). If it matches (BAM has correctly solved the task), the environment sends only this pattern back to the FEBAM (then to the BAM).

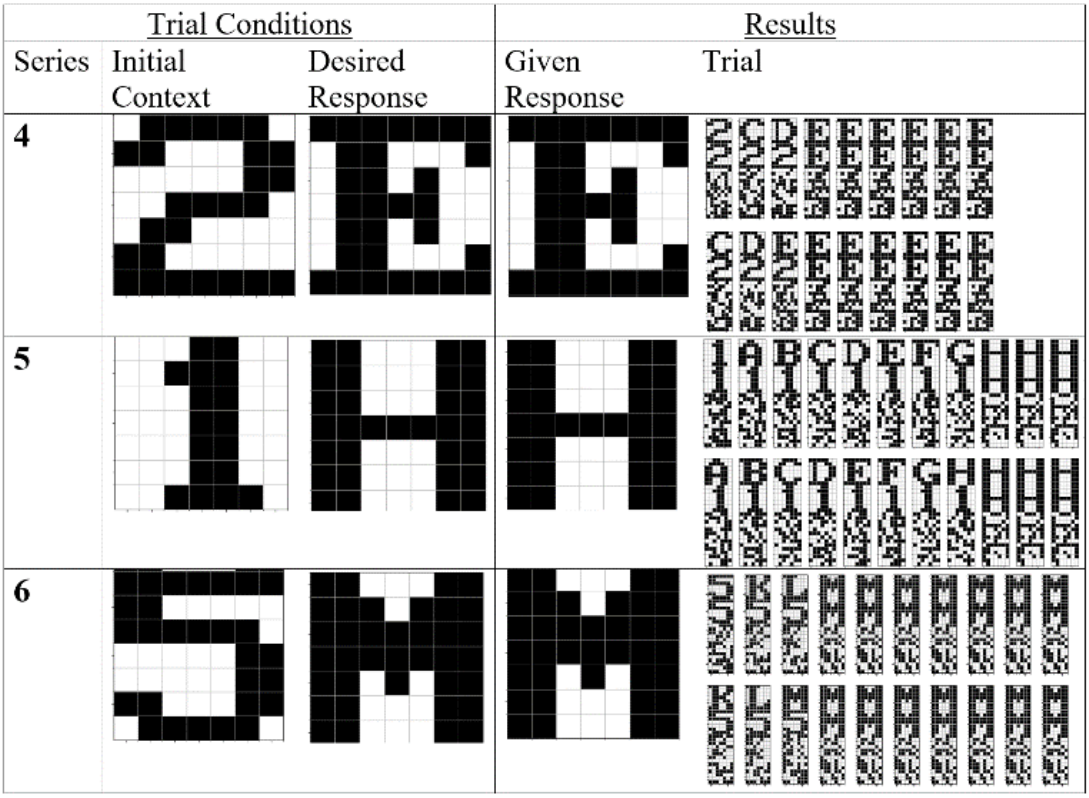
5. The BAM recalls the pattern associated with itself and stabilizes instead of continuing to iterate through the series.

Results

As shown in Figure 12, the FEBAM-BAM model was able to learn and perfectly recall both time series 4 and 5 with no problems, which the BAM alone was not able to do. It could also stabilize on any given response without issue. In the example shown for series 4 (Fig. 12), the desired response for that trial was ‘E’ in the ‘2’ time series. The first input for that time series, ‘2’, was given as the initial input and sent through the FEBAM, where the unique signature for that pattern was generated and appended to it. This new stimulus was then entered as input for the BAM, which gave ‘C’ as an output. That output was compared to the desired response in the environment and was sent back to the FEBAM until the desired response, ‘E’ was given. This desired response ‘E’ was then sent, associated with itself, as the new input for FEBAM, causing the BAM to stabilize on the correct answer. Most importantly, the model was able to successfully stabilize on any given response in series 6, which contained all possible combinations of interaction and correlation between stimuli: subtype, independent time series, and overlap. This model was successfully able to iterate through all series and stabilize on a given response.

Figure 12

Results of Example Trials from Simulation II



Discussion and Conclusion

To further our understanding of Reinforcement Learning and the exploration-exploitation trade-off, the exploitation phase was implemented using a BAM. Here it was assumed that a given set of patterns were already encoded for various contexts. The network's job was then to iterate through a given series and stabilize on the desired response using feedback from the environment.

Various levels of complexity were tested ranging from a single time series all the way to the same variability found in a natural setting. When the complexity was too difficult for a single

BAM (nonlinearly separable cases), a FEBAM was included in order to generate unique representations for each pattern. This combination of networks was sufficient to solve any type of situation.

Furthermore, by adding layers of FEBAM-BAM, the point attractors could become the context of a new time series and allow chains of time series; something akin to chunking (Bhandari & Duncan, 2014; Gobet et al., 2001).

One limitation of this study is that it represents exploitation under very specific circumstances. For example, if one is trying to solve a problem with a series of learned responses, they will learn which of the responses are helpful and which are not. Therefore, the order of the patterns should reflect the probability of success, a phenomenon well-captured by standard Q-Learning. Changing the order of some items without retraining the whole dataset is a challenging avenue in a distributed associative memory and should be addressed in future studies. This could be implemented using an additional BAM to store "correct" responses in function of the desired new order with a novel correlated context generated by the network itself or by switching to more interesting attractors such as aperiodic ones (Tsuda, 2001).

The current network could also be added to a model of exploration to study the exploration-exploitation trade-off. Finally, temporal aspects should be taken into consideration, allowing for the timing of when actions should be accomplished, easing the transition from numerical simulation to real-time neurorobotic implementation.

In conclusion, by using a BAM, we were able to model the exploitation phase of the exploration-exploitation dilemma, where the subject iterates through different learned responses and can stabilize on the correct response based on feedback from the environment. By adding a

FEBAM to generate a unique representation for each learned stimulus, we were able to model this with all of the complexity of a real-world setting, including different lengths of stimuli, different levels of correlation, and nonlinear problems. Being able to model exploitation is a crucial part of understanding our own cognition and how we learn from the dynamic world around us.

CHAPTER 3: Study 2 - Generating Distributed Randomness using Artificial Neural Networks

Suppose you are asked to choose randomly between left or right 100 times, would you expect the average of your choices to be roughly even or to have a bias? In the literature, human randomness falls on a spectrum from being close to unbiased to very biased in random choices. To create a model with a neural implementation of human randomness, unsupervised artificial neural networks were used to generate a random representation of binary numbers. These random representations were tested with both orthogonal and correlated stimuli as inputs and the properties of all outputs are discussed. An example of how to bias this generated randomness to model different cognitive processes is shown under two conditions, where random decisions are biased for desired outcomes and for list exhaustion (random sampling without replacement). Other possible uses for this method of generating randomness in cognitive modelling are discussed.

Keywords: distributed randomness; biasing decisions; feature extraction bidirectional associative memory; artificial neural networks; pseudo-random number generation

Introduction

If you were asked to pick a number from one to ten, which would you choose? Is this decision completely random or do you tend to prefer a certain number? What if you had to choose 100 times, does this change anything? A certain level of randomness is present in all our decisions, from deciding whether to bet on red or black at the roulette wheel to making large and impactful life decisions. However, there is still a lot of ambiguity about the definition of the concept of randomness (Nickerson, 2002). Jokar and Mikaili (2012) write that “[a] sequence of

numbers is said to be random, if the next element cannot be predicted from the previous one." In our study, randomness is also viewed in terms of unpredictability.

There is a variety of contrasting literature on human randomness, especially concerning how much bias is in a random decision. Several systematic reviews of human randomness research have found that most studies suggest humans have difficulty generating random sequences or recognizing random patterns. However, some argue that findings of bias in randomness could be the result of ambiguous instructions, issues with the statistical methodology, or the participant's limited window of experience (Ayton, Hunt & Wright, 1989; Nickerson, 2002; Warren et al., 2018). Others argue that under certain conditions, it could be possible for human randomness to be almost entirely unbiased by instructing participants differently (Guseva et al., 2023).

On the other hand, there are studies that do provide evidence that human decisions are not truly random and that there is a bias. One study shows how our perception of patterns can lead us to fall for the gambler's fallacy and the hot hand when watching videos of other people gambling (Croson & Sundali, 2005). Another study looked at how choices are considered using both behavioral data and neuroimaging techniques and found that when choosing between items, participants chose items from a preferred category more often and more quickly (Lopez-Parsem, Domenech & Pessiglione, 2016). A different study found that since random numbers do not meet the full randomness criteria, distinctive features in response patterns can be used to identify which subject produced what random value, providing more evidence that humans do not follow statistical randomness (Jokar et al., 2012).

Modelling Cognition and Randomness with Artificial Neural Networks

Many Artificial Neural Networks (ANNs) that model cognition rely on randomness to be able to illustrate the arbitrary nature of neural coding in the brain (Kanerva, 2009). To generate this pseudo-randomness, a lot of popular approaches either forego the use of neural networks (Blum, Shub & Blum, 1986; Matsumoto & Nishimura, 1998) or generate a numerical value as output (Malik, Pulikkotil, & Sharma, 2021). Those that do generate pseudo-random sequences use less biologically plausible models such as deep neural networks (Almardeny et al., 2022; Park et al., 2022; Pasqualini & Parton, 2020).

The approaches that use recurrent neural networks rarely offer a complete neural network-based solution and are often combined with other data sources or algorithms (Jeong et al., 2018; Man et al., 2021; Tirdad & Sadeghian, 2010). Similar approaches rely on random orthogonal weight matrices (Elyada & Horn, 2005; Hughes, 2007), exploit an input source of randomness (De Bernardi, Khouzani & Malacaria, 2019), or use backpropagation (Desai, Ravindra & Rao, 2012), methods that are less compatible with cognitive modeling. Our model is an attempt to merge this gap in the research by generating pseudo-random sequences, or distributed representation of randomness, entirely using recurrent neural memory models.

Two sets of simulations were conducted. The first aims to evaluate the capacity of an ANN to generate distributed random behaviors and the second to bias those random representations.

Background

The ANN used in this study was the Feature Extracting Bidirectional Associative Memory (FEBAM; see Chartier et al., 2007 for more details). The FEBAM is an unsupervised memory

model that generates a unique representation of each input learned, which can then be used in many ways. For example, these generated representations have previously been used as a "unique signature" or identifying representation for different inputs (Church, Ross & Chartier, 2020; T. Rolon-Mérette, D. Rolon-Mérette & Chartier, 2018).

The FEBAM was selected for this unique property, as we hypothesize these generated representations could be adapted to represent different aspects of randomness, such as the likelihood of making a given decision. Using a binary representation means that the outputs can represent a quantity and the sum of the outputs can be calculated with a single neuron. As with any artificial neural network model, the FEBAM can be entirely described by its architecture, transmission and learning functions.

Architecture

The FEBAM architecture is comprised of two layers of units which are interconnected in a bidirectional fashion where $x_{[0]}$and $y_{[0]}$ are the initial inputs (Fig. 13). The weight matrices, W and V, connect the network units, x and y, and return information to each other. The inputs $x_{[c]}$ and $y_{[c]}$ are the inputs at the current iteration number c.

Transmission

The transmission used a binary function of the original cubic one (T. Rolon-Mérette, D. Rolon-Mérette & Chartier, 2023). It is expressed by:

$$\forall i, \dots, m, \boldsymbol{y}_{i[\mathrm{c}+1]} = \begin{cases} 1, if\ \mathbf{W}\mathbf{x}_{i[\mathrm{c}]} > 1 \\ 0, if\ \mathbf{W}\mathbf{x}_{i[\mathrm{c}]} < 0 \\ 3(\mathbf{W}\mathbf{x}_{i[\mathrm{c}]})^{\mathbf{2}} - 2(\mathbf{W}\mathbf{x}_{i[c]})^{3}, Else \end{cases} \tag{5}$$

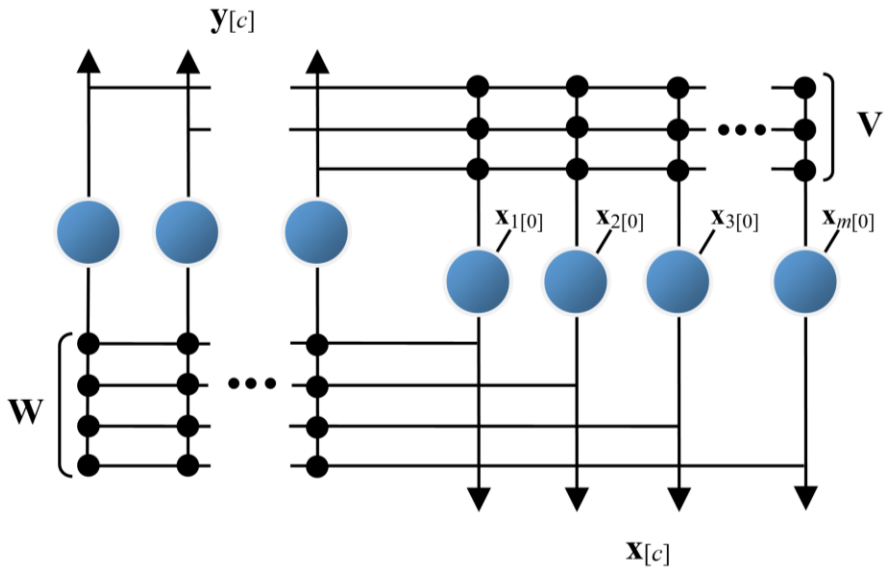
Where m is the number of units in each layer, i is the index unit, and c is the cycle index; for all simulations it was set to 1. A similar process is used to obtain $\mathbf{x}_{i[c+1]}$ by replacing $\mathbf{W}\mathbf{x}_{i[c]}$ with $\mathbf{V}\mathbf{y}_{i[c]}$.

Learning

The learning is based on time difference Hebbian learning**,** as seen in Equations 3a and 3b in Chapter 2.

Figure 13

Architecture of the FEBAM



Simulation I – Properties of Generated Randomness

As discussed in the literature, while randomness is on a spectrum, it may be possible for human randomness to be almost entirely unbiased under specific instructions (Ayton et al., 1989; Nickerson, 2002; Warren et al, 2018). Thus, the first step was to make sure that regardless of the input, it was possible for the network to reliably generate a random output; a process akin to computer pseudo-random generators. More precisely, we studied the capacity of the network to generate values between 0 and 1. If the proportion of 1s and 0s are equal, we would get an unbiased binomial distribution. For the latter, a sum of greater than 50% could represent decision A and less than 50%, decision B.

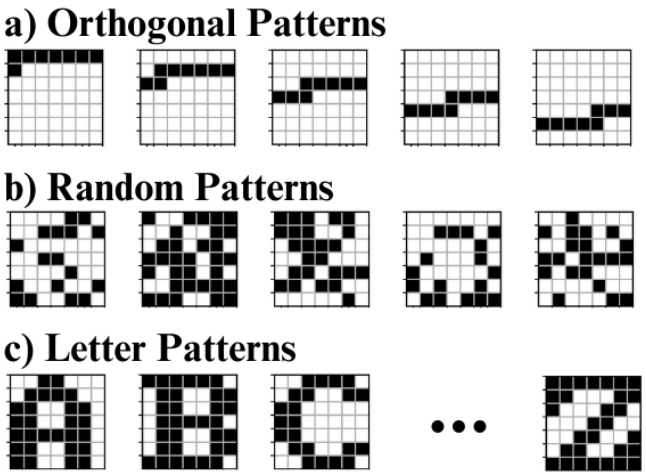
Method

Inputs. Three different sets of inputs were used: orthogonal, randomly generated (correlated) and uppercase letter (correlated) patterns were tested to evaluate their effect on the generated patterns (Fig 14). For comparison purposes, they were all the same size of 7x7 pixels, giving 49 dimensions. Orthogonal and random patterns were also tested with higher dimensionality. Black pixels were assigned a value of 1 and white pixels a value of 0.

Parameters. To assess if the dimensionality, m, of the generated representation had an impact on its variability, the number of y-units varied from 49 to 196. The weights were initialized between [-0.1, 0.1] for each trial and the learning rate (η) was set to $1/m$. Finally, in some conditions, binary discretization of the outputs was performed. The error was calculated by measuring the average of the squares of the errors, Mean Squared Error (MSE), and minimum MSE was set to 10^{-8}.

Figure 14

Examples of Different Input Patterns of 49 Dimensions



Learning Procedure. The learning was accomplished as follows:

1. Selection of input patterns (Fig. 14)

2. Computation of the output (Eq. 5) and update of the weights (Eq. 3a and 3b).

3. Repetition of 1-2 until the minimum MSE is reached.

Orthogonal Patterns

The network was first tested under optimal patterns condition offered by orthogonality. By using such patterns, it controls any interactions the patterns may have that would be reflected on the generated representation by the network. This situation is closer to the condition in which pseudo-random generator is used, where the current draw is independent from the previous. In addition, orthogonal inputs can be used to represent one-hot encoding (Cohen et al., 2013), converting categorical data into numerical data, which is still commonly used today (Park et al., 2023; Ranasinghe et al., 2021). More precisely, 250-dimension orthogonal patterns with different

levels of active pixels (value of 1) were tested (1, 2, 5, 10, 25, 50, 100). See Figure 14a for an example of 8-pixel 49-dimension patterns. The number of patterns learned varied from 1 up to 50. Each condition was repeated 100 times and average performances were reported.

Finally, the type of distribution generated by the network was assessed. Thus, the network was trained using *n*-orthogonal patterns. Recall was performed where the output with the maximum sum was selected. Learning was then repeated for 200 trials to have a high sample. A chi-square was applied to measure the departure of the final distribution with the expected multinomial one.

Results. As shown in Figure 15, the distribution of orthogonal inputs across 200 trials is not different than the theoretical uniform distribution either for the binomial, $n = 2$, χ^2 (1, 199) = 0.09, $p = 0.76$ (Fig. 15a) or the multinomial one, $n = 5$, χ^2 (1, 199) = 2.19, $p = 0.7$ (Fig. 15b).

Additionally, these trials showed that with fully orthogonal patterns, the number of pixels in the input had a large effect on the average summation percentage of the random outputs (see Fig. 16). The more pixels in the input pattern, the higher the sum. We tested this with higher dimensionality and more input patterns, and the trend continued. With the binary discretization applied (threshold), the resulting outputs are around 50%, but this increases with the dimensionality of the input patterns, so it is not stable.

Figure 15

Binomial and Multinomial Distribution of Orthogonal Patterns, with the Expected Percentage

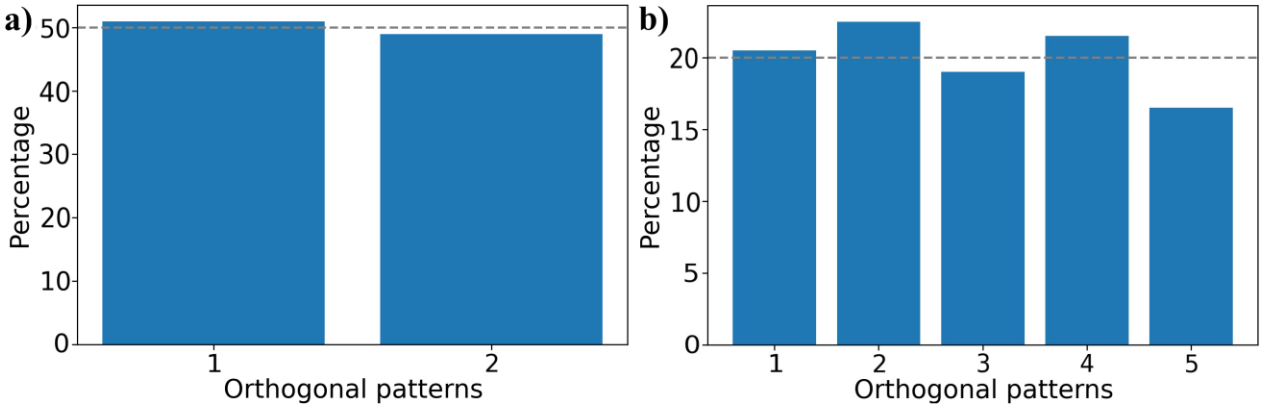
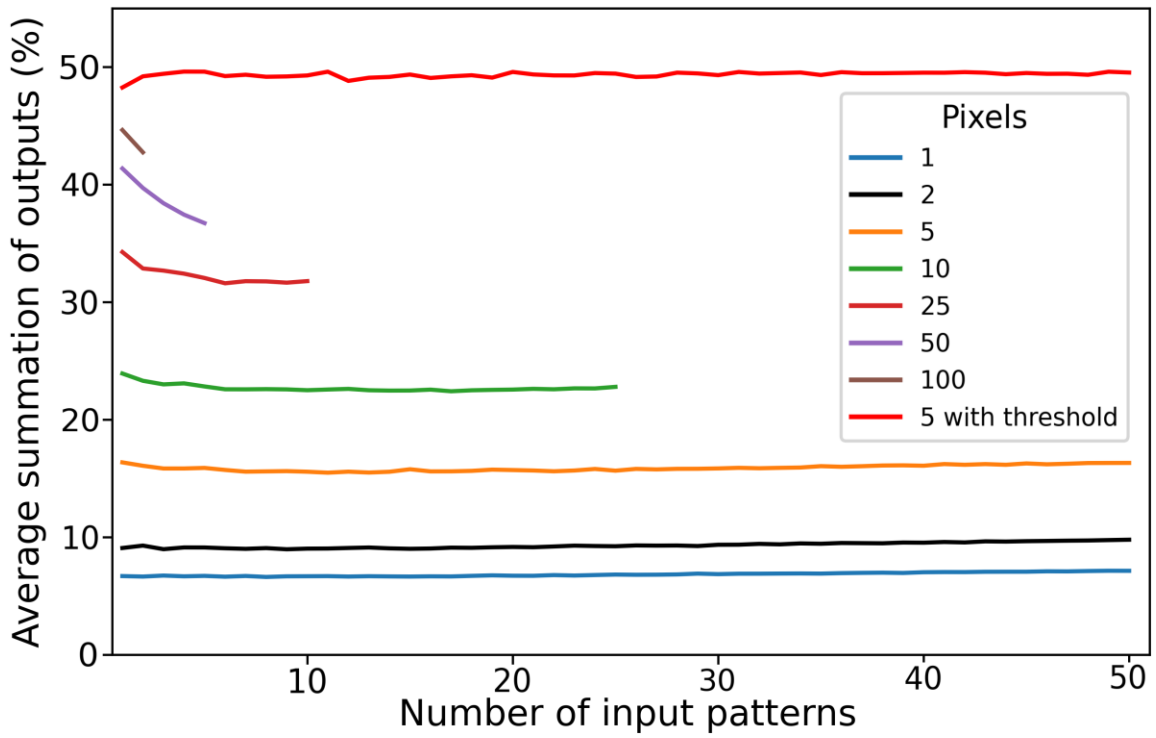


Figure 16

Orthogonal Patterns with Different Number of Active Pixels in Input Patterns



Random and Structured Correlated Patterns

In a more natural setting, input patterns are correlated; whether randomly generated (Fig. 14b) or structured (like uppercase letters; Fig. 14c). The binary random patterns were randomly generated from a discrete uniform distribution over [0,1], resulting in an average of 50% active pixels per pattern. Like the previous section, the number of inputs patterns was varied to assess the behavior under various memory loads and the distribution was assessed by a chi-square.

Results. When using structured patterns, the resulting distribution with the highest sum is not uniform χ^2 (1, 200) = 0.09, p = 0.76, as shown in Figure 17. The number of times the letter "B" is selected is significantly higher than the other letters. This is not surprising since the letter "B" has the smallest Euclidian distance from the average of all the five letters.

Another major difference from the orthogonal trials is that the number of learned inputs has an effect on the average sum of the generated patterns. As the number of input patterns increases, the raw summation of the outputs increases (see Fig. 18). The observed generated behavior depends on the number of patterns learned and not the memory load itself. For example, 10 patterns of 98 dimensions (memory load of 10%) will have around the same proportion of 1s as 10 patterns of 196 dimensions (5%). As the sum depends on several different factors like the dimensionality of the inputs, number of inputs, and use of the threshold function, it is difficult to determine stable relationships between these variables to have a predictable outcome. There was no difference between the random correlated patterns and the structured correlated (uppercase letter) patterns.

Figure 17

Distribution of Letter Patterns, with the Expected Percentage

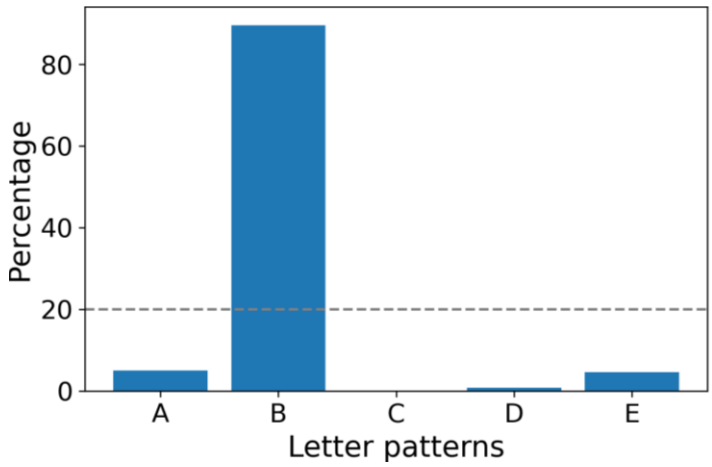
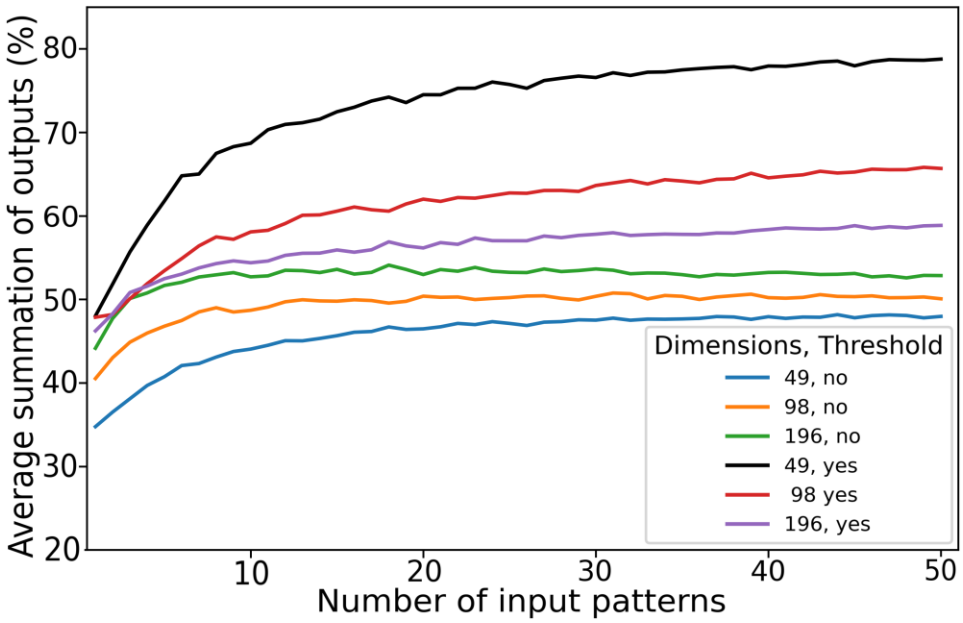


Figure 18

Random Inputs with and without the Threshold Function Applied, 3 Different Dimensionalities



Discussion

The results from the orthogonal patterns task confirmed the hypothesis that by using distributed representations in an ANN, it is possible to create uniformly distributed random representations and as such could replace pseudo-random generator when modeling cognitive processes such as decision processes. This process is akin to sampling with replacement. For example, if we want a system to perform Task X 80% of the time and Task Y 20% of time, one can use 5 orthogonal patterns where patterns 1 to 4 will represent Task X and pattern 5 will represent Task Y (Figure 15b).

However, with the correlated pattern trials, the results were more varied. With certain dimensions and only a single letter as the input pattern, it is possible to have an equal proportion of 1s and 0s. When more input patterns are used, the threshold will induce a bias towards 1, while the straight sum towards 0. This is not ideal for many situations where more than one input is used. Some patterns are more likely to be chosen than others because they have a smaller Euclidean distance from the average of all patterns than others.

On the other hand, sometimes a biased distribution is exactly what is desired. This can be seen as a better representation of random decisions in the real world, where we may be inherently more likely to be biased towards certain decisions even before any learning has taken place, based on our attention (Orquin & Loose, 2013), what our goals are and the salience of the stimuli (Berridge & Robinson, 2003), or some other internal or external bias (Wittmann et al., 2008).

As would occur in the real world, we can always change our behaviors, regardless of our original internal biases. The next simulation presents such implementation of how decision can be modified even under initial bias.

Simulation II – Biasing the Random Outputs

As seen in Simulation I, when orthogonal input patterns are used, the network can give a series of uniformly distributed output patterns (sampling *with* replacement). In some situations, we would like to have the network perform sampling *without* replacement, where the network could exhaust a list of patterns without repeating any previous one.

Another situation could be that it may not be possible to use orthogonal patterns. In the real world, it is more likely that there will be some overlap between possible decisions to be made. As seen in Figure 17, this results in a biased distribution and can be seen as analogous to our internal biases, where we are more likely to select some decisions over others. In both situations, by introducing a selection parameter on the output, it will be possible to generate time series and modify any initial bias. These two conditions were tested in Simulation II: Desired Output and Sampling without Replacement.

Desired Output Condition

For this condition, a series of correlated letter patterns (Fig. 14c) are learned by the FEBAM. These input patterns represent different possible decisions/behaviors. The desired decision is established ahead of time. During each trial, the learned letter inputs are sent for recall and the FEBAM recalls the corresponding random representation for each letter. The sum is then taken from each of these patterns and the one with the highest sum is “chosen”. If we want a different outcome, we simply multiply the output by a value between [0 and 1[(ex. 0.1)

before computing the sums. The effect of this selection parameter ($\boldsymbol{\beta}$) can be expressed by the following:

$$\mathbf{o}_i = \boldsymbol{\beta}_i \sum_j \boldsymbol{y}_{ij} \tag{6a}$$

$$\mathrm{s} = \mathrm{Max}(\mathbf{o}_i) \tag{6b}$$

Where $\mathbf{y}_i$ represents the output pattern i, $\boldsymbol{\beta}$, the selection parameter, j the given pixel ($j \ldots 49$), $\mathbf{o}$, the outputs of all patterns, and s, the resulting selected pattern.

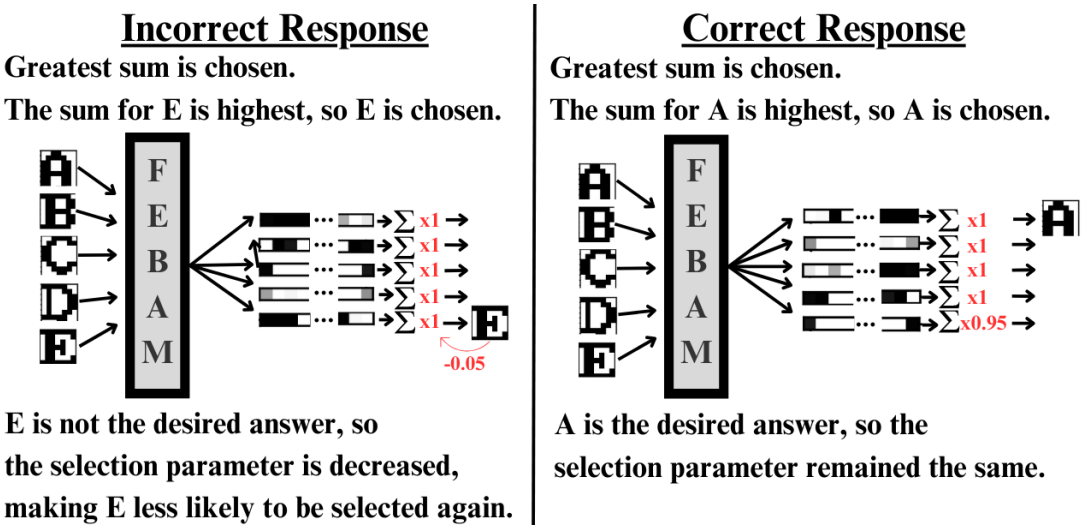
This selection parameter is tailored for each letter. The values of $\boldsymbol{\beta}$ are initialized at 1 for each pattern (which will have no effect on the first decision). After each recall, the selection parameter can be decreased if needed by a constant, ε. Finally, the parameter ($\boldsymbol{\beta}$) has a lower limit at 0, which results in removing this particular choice as a possible outcome. The strength, ε, of the gain can be altered to change how fast or slow the model changes responses. For the condition of correlated pattern ε can be set to a value closer to 0 for slow transition:

$$\boldsymbol{\beta}_{i[t+1]} = \begin{cases} 0, \ if \ (\boldsymbol{\beta}_{i[t]} - \varepsilon) < 0 \\ \boldsymbol{\beta}_{i[t+1]}, Else \end{cases} \tag{7}$$

For example, if we have an input list “A, B, C, D, E” and the desired letter is A, the network will output representations for each letter. The one with the highest sum will be selected. If the corresponding letter is any but A (such as E), the gain value will be decreased. Therefore, for the next recall the sum of E should be lowered until a new letter is selected (see Fig. 19, “Incorrect Response”). When the correct output is selected, then the selection parameter remains the same (see Fig. 19, “Correct Response”).

Figure 19

Flowchart for the Desired Output Condition



Parameters and Learning Procedure. The network parameters and learning procedure were the same as in Simulation I. A series of up to 10 correlated letter patterns were selected (Fig. 14c). Various levels of gain strength were tested, and a value of (ε=) 0.05 was set for slow and (ε=) 0.5 for fast response changing. The effect of the selection parameter was investigated while finding the desired output. Each condition was tested using different subsets of letters.

General Selection Procedure.

1. Learning of the patterns according to the procedure in Simulation I.

2. Each learned letter pattern is sent to the FEBAM to obtain its corresponding representation (Eq. 5).

3. Each representation is then modified using Eq. 6a and 6b to select the letter associated with the largest sum.

4. If that chosen letter is the desired one, then the selection parameter remains the same, if not, it is decreased according to Eq. 7.

5. 2-4 are repeated until the maximum number of trials has been reached.

Results. The results show that it is possible to bias towards a certain letter. By using the selection parameter, each time a letter is chosen its corresponding sum may be modified, allowing for the desired letter to be chosen repeatedly. When the desired letter is changed, the network is able to inhibit the recall of the previous letter and move to the next one until it chooses the correct one.

The strength of the gain on the selection parameter has an impact on how many trials it takes to stabilize on the correct response. For example, if the desired output was letter A followed by letter C, when the selection parameter was decreased by a small amount (0.05; Fig. 20a), it takes almost 11 trials for selecting A. Conversely, when the selection parameter is decreased with greater quantity (0.5; Fig. 20b) it took only 4 trials for selecting A. It was successful in both conditions. It is also possible for the network to restabilize on previously learned letters, as seen in Figure 21.

Each time a different decision is made it will slowly decrease the selection values. Therefore, at some point the values will all be zero and the network will then always select the first letter in the series (see Fig 22a). One can always reset the parameter to initial values ($\boldsymbol{\beta} = \mathbf{1}$) when $\boldsymbol{\beta} = \mathbf{0}$ and start a novel decision stream. However, if the decrease in the selection parameter is slow, it will have enough trials to exhaust the list of long time series, even with a lot of fluctuations (see Fig. 22b).

Figure 20

Difference in Slow vs. Fast Selection Parameter for Desired Letter "A" for 25 Trials and "C" for 25 Trials

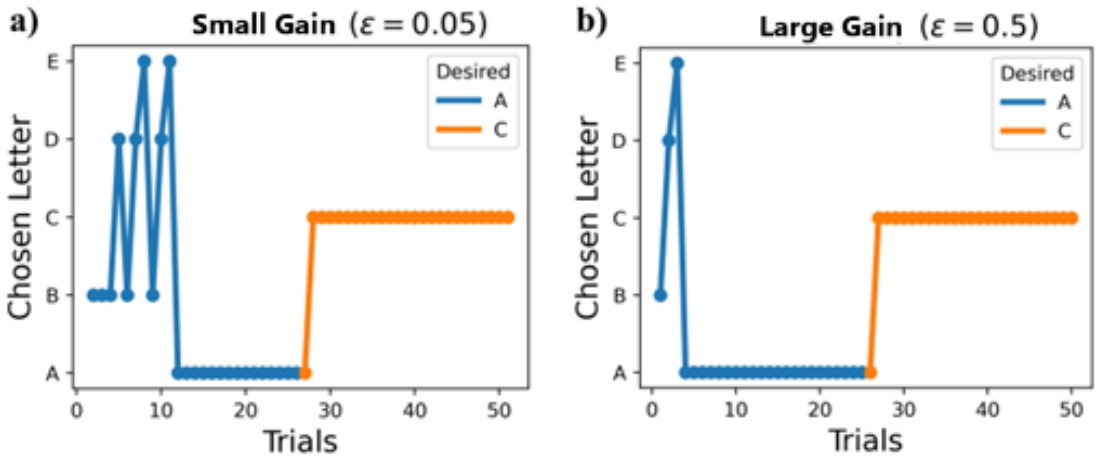


Figure 21

Chosen Letter over 100 Trials, where the Desired Letter is "F" for 25 trials, then "G", "I", and "G" again

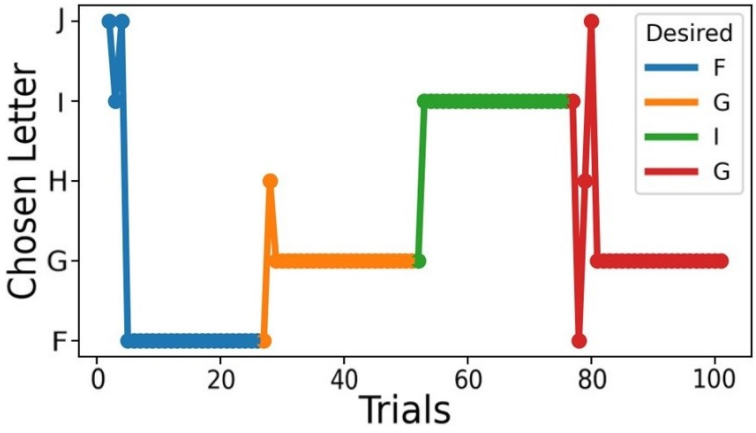
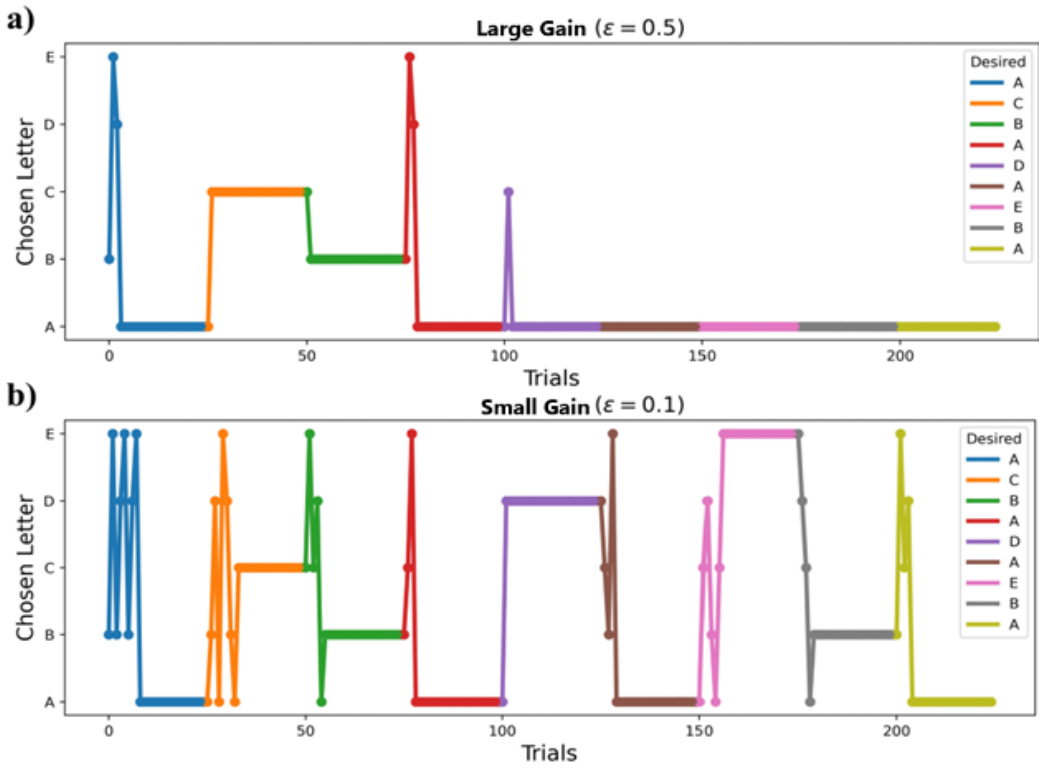


Figure 22

Chosen Letter over 250 Trials where the Desired Letter Changes Every 25 Trials

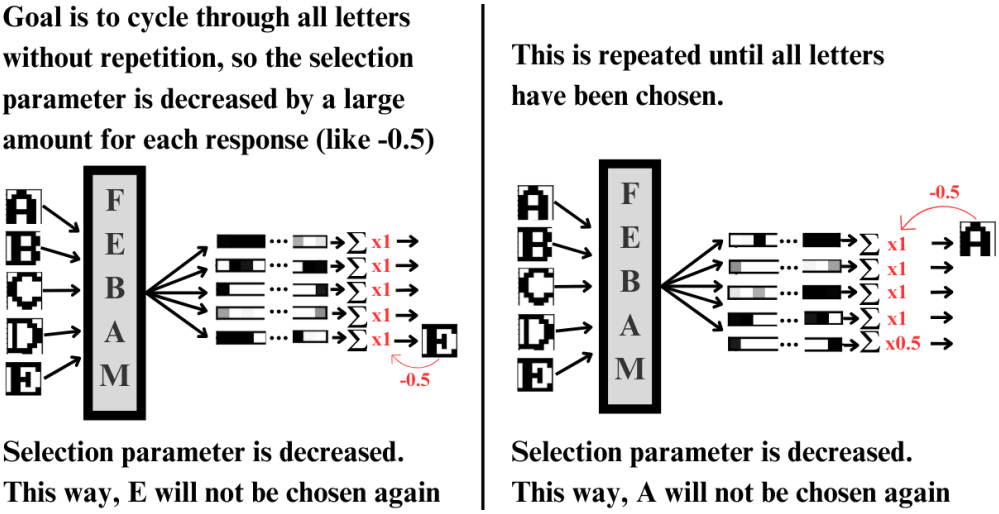


Sampling without Replacement Condition

For this condition, a series of orthogonal or correlated letter patterns are learned by the FEBAM (Fig. 14a and 14c). The goal is to select each of these patterns once, which is akin to list exhaustion or random draw without replacement. Since each pattern should be selected only once, ε will be set higher than in the previous condition (between 0.5 and 1) to ensure that selected patterns will not be selected again (Fig. 23).

Figure 23

Flowchart for the Sampling without Replacement Condition



Parameters and Learning Procedure. The network parameters and learning procedure were the same as in Simulation I. A series of 4 orthogonal (composed of 10 active pixels) patterns or correlated letters were chosen. The effect of the selection parameter was investigated while changing the value after the learning has been accomplished. Each condition was tested using different subsets of letters.

General Selection Procedure.

1-3. Same as the Desired Output Condition.

4. The selection parameter is decreased according to Eq. 4.

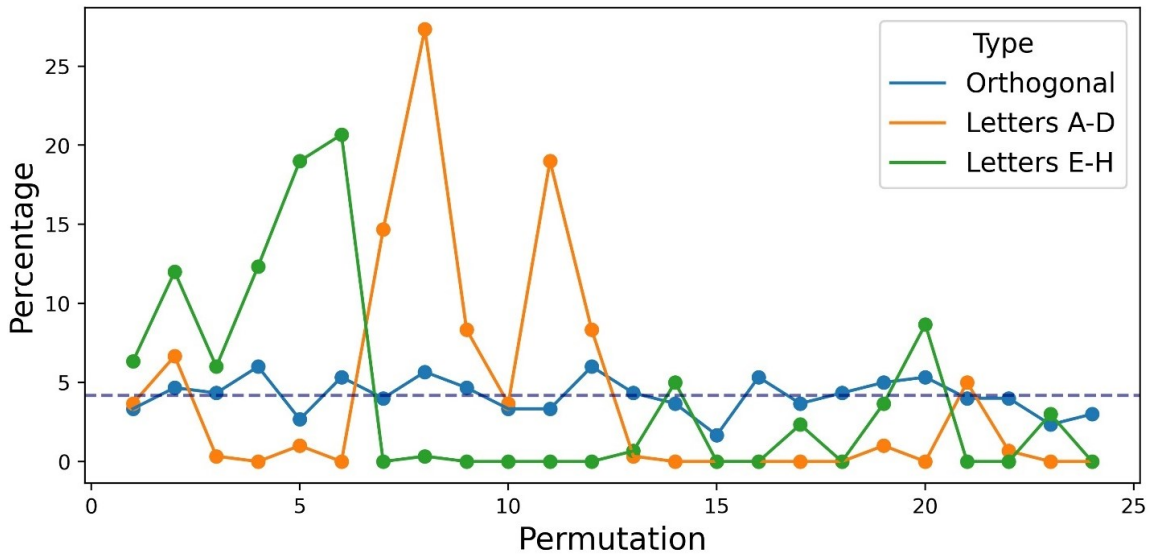
5. 2-4 is repeated until all patterns have been selected.

6. 1-5 is repeated 300 times and percentages are computed.

Results. With 4 patterns there are a total of 24 (= 4!) different permutations: 1 = [1 2 3 4], 2 = [1 2 4 3], 3 = [1 3 2 4], …, 24 = [4 3 2 1]. The frequency of each permutation was computed (Fig. 24) and revealed no difference with the theoretical one of $4.1\bar{6}\%$ (= 100/24) for the orthogonal patterns (χ^2 (1, 299) = 13.92, p = 0.93). Which is not the case with the correlated letter patterns, whatever the combination (A to D or E to H), some permutations have a higher probability of occurrence than others (Fig. 24), as seen in Simulation I. In both situations, we observed statistical difference (A-D: χ^2 (1, 299) = 824, $p < 0.01$; E-H: χ^2 (1, 299) = 628, $p <$ 0.01).

Figure 24

Orthogonal and Letter Sampling without Replacement Trials with Theoretical Frequency



Discussion

When orthogonal patterns are combined with FEBAM and the selection parameter the results showed that random permutation can be obtained. Each time series will have the same probability of being selected, making the lists independent and identically distributed (IID). Of course, if a particular sequence is desired it is possible for the network to extract it in the right order. This was shown using a biased output obtained when using correlated patterns.

Moreover, results show that even if there is a preference in the output (ex. letter B), it is possible to circumvent this bias and output the desired behaviour. This provides a possible neural implementation that supports the studies which argue that there is bias in human randomness (Croson et al., 2005; Jokar et al., 2012; Lopez-Parsem et al., 2016).

We show that the network can also produce a series of outputs with repeating patterns, where the length of the time series is determined by the strength of the selection parameter. This bias can shift over time to a different choice to represent changes in internal bias. The network could even be tailored to individual preferences by initializing the selection parameter matrix for each choice to something other than 1. It could also be modified in function of the task.

In future work, this more plausible implementation of human randomness could be integrated with different cognitive models for learning, creativity and problem solving. For example, by combining this with an existing neural model for creativity, it could allow the model to break away from certain patterns, resulting in more creative decisions and more improvisation (Khalil & Moustafa, 2022). Using randomness can help us break free from fixed patterns of thinking to discover novel solutions. Thus, it is important to include an accurate model of human randomness when modelling any of these cognitive processes.

This ANN model for pseudo-random sequence generation adds another option to existing recurrent neural network methods that does not rely on the addition of external algorithms (Jeong et al., 2018; Man et al., 2021; Tirdad et al., 2010), orthogonal weight matrix initialization (Elyada et al., 2005; Hughes, 2007), randomized inputs (De Bernardi et al., 2019), or backpropagation (Desai et al., 2012). It also offers an alternative to the deep neural networks that are commonly used in reinforcement learning (Almardeny et al., 2022; Park et al., 2022; Pasqualini et al., 2020). As an added property, our model was able to go beyond single digits and generate distributed arrays of randomness.

Conclusion

This study provides a more accurate implementation of human randomness using an ANN. The results suggest that the model can generate unbiased distributed random representations when orthogonal inputs are used. The model can be biased towards different choices, which allows us to model human randomness even when it is not as random as it may seem. This more biologically plausible representation of internal randomness could be used to create more realistic cognitive models of random and biased decisions.

CHAPTER 4: General Discussion

Overall Findings

Study 1 showed that it is possible to implement exploitation in a BAM, assuming that the responses to be exploited for each task were already learned. Using context to determine the task, the network was able to iterate through a given series and stabilize on the desired response using feedback from the environment. When the complexity of the task was too difficult for a single BAM (for nonlinearly separable problems), a FEBAM was used to generate unique representations for each pattern, creating an architecture that was able to model pure exploitation, regardless of the level of variability in the task.

Study 2 presents a new way of generating randomness with some properties that could be useful in different types of computational modelling. Namely, it shows a method for generating randomness that is distributed across an entire pattern/matrix. This model can represent randomness in all different kinds of decision making, situations, and tasks. For example, the letter stimuli used as inputs can represent almost anything, from visual inputs to a series of motor actions where each pixel represents the presence or absence of a behaviour, to more abstract concepts like complex actions or thoughts. It is also easy to switch between biased and unbiased randomness by modifying the inputs (orthogonal or correlated), the strength of the gain, or the selection parameters. These features make it very easy to adapt these models to represent a wide range of tasks in cognition and beyond.

In the first study, only bipolar inputs were used (-1 and +1) to represent information. In the second study, we switched to binary inputs (0 and 1) to represent information because of the added properties it provided. Namely, it allowed us to easily sum the number of active pixels using a single neuron and it does a better job of representing the sparseness seen in real neurons.

All simulations from Study 1 have since been replicated using binary patterns and the modified neural networks to achieve the same results. With this modified binary version of Study 1, both networks are now compatible and will be able to be combined seamlessly for future studies.

Future Research

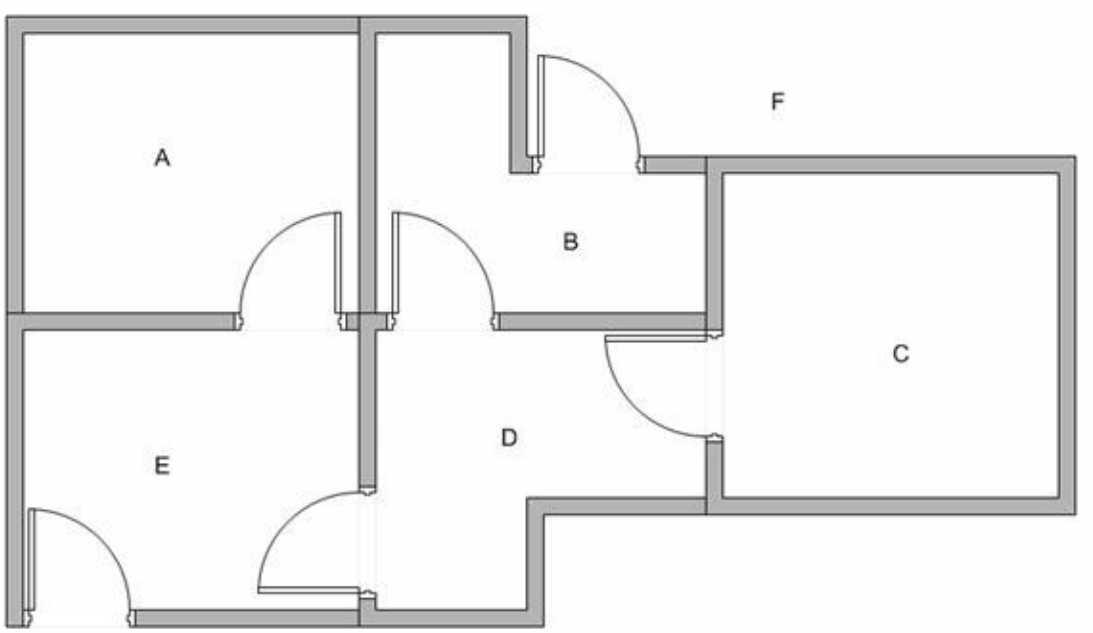
Preliminary results for a basic maze task implementation

The exploration model from Study 2 can be combined with the exploitation model from Study 1 and used in an applied context, like the maze task. As a proof of concept, this was tested on the basic maze used in Teknomo (2019) to see how it would explore to exit the maze and store that information to be exploited later on. This maze is comprised of six possible states (A-F), with F being the desired state, where the agent has exited the maze (see Fig. 25).

The main goal of this pilot test was to see if the random generation FEBAM from Study 2 can generate the randomness needed to explore and eventually exit the maze. Instead of receiving a reward for each correct action like in Q-Learning, this model uses the selection parameters from Study 2 with a large gain (ε=0.5) so that the incorrect pathways can only be selected a maximum of two times before a different pathway is selected (the selection parameter for that choice is reduced to 0 after being chosen twice). Rather than storing information about the optimal actions in a Q-table reward matrix like in Q-Learning, this implementation stores the optimal decisions from each room in a separate BAM, which can be used to exploit learned information later on and exit the maze flawlessly, just like in Study 1. Rather than selecting from available choices randomly like in Q-Learning, this implementation uses the random FEBAM from Study 2 to exhaust the list of available choices until the response is rewarded, which is advantageous because it will take less trials to learn the task.

Figure 25

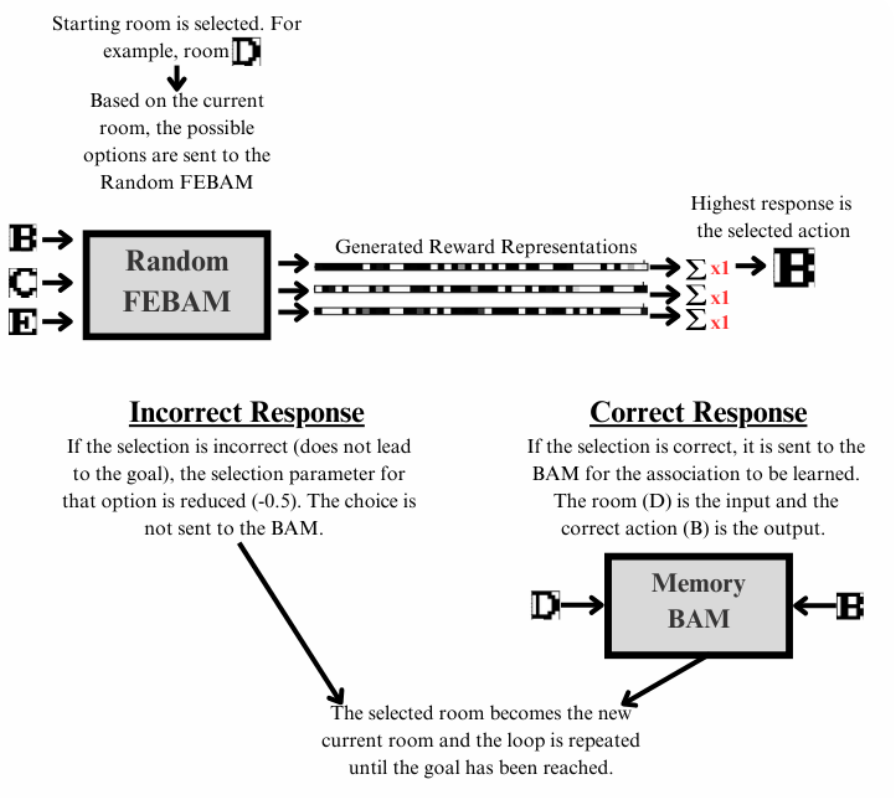
Basic Q-Learning maze from Teknomo (2019) used to test the model from Study 2



This maze offers a one-to-many problem, where the agent must decide between two equally valid options (D>B>F vs. D>E>F). In theory, the randomness generator from Study 2 would choose room "B" more often than room "E" because "B" has the smallest Euclidean distance from the average of all of the letter patterns, as discussed in Study 2. This can either be seen as a feature or a bug. In the real world, some pathways are more likely to be selected in general based on our attention (Orquin & Loose, 2013), our goals and the salience of the stimuli (Berridge & Robinson, 2003), or some other internal or external bias (Wittmann et al., 2008). For this trial, one option will be selected as "correct" to facilitate the learning of time series in the BAM. Future implementations could investigate other ways of solving this one-to-many problem.

Figure 26

Flowchart of the model used to pilot test on the maze task



A flowchart of this implementation is shown in Figure 26, where the agent's current room will determine what inputs and selection parameters are used. For example, if the current room is "D," the next possible actions are rooms "B", "C", and "E." These are the inputs for the random FEBAM. The selection parameters are set to [1,1,1] to start, so the generated representation of each option will be multiplied by 1. In this case, room "B" is selected and becomes the new current room. Since "B" is a correct room and leads towards the goal, the selection parameters are not modified. If the model had selected "C," which does not lead to the goal, the selection parameter for "C" would be reduced by 0.5, meaning that the maximum number of times the network can revisit the same choice is 2. This makes it unlikely to be

selected in the future. If we wanted to eliminate the possibility of selecting the wrong room twice, we could increase the gain to 1. This way, the selection parameter would be reduced to 0 after one choice and it would be impossible for the network to revisit that option again.

Once the correct response has been selected, it can then be fed into the memory BAM in the same manner as it was learned in Study 1 (associated with the starting state as seen in Fig. 26). These learned associations can later be used to exit the maze. This method will work even under noisy conditions.

Results showed that the random FEBAM was able to randomly select from the available options for each room. After making an incorrect decision, the model would not repeat the mistake by visiting that room more than twice. It was able to exit the maze successfully each time. See Figure 27 for some examples of pathways to exit from starting room “A”. After exiting the maze, the memory BAM will have learned the time series for how to exit the maze from that starting room (see Fig. 28a). If desired, context could be used to learn how to exit from different starting rooms, or how to navigate to different goal rooms, by simply appending the context to each action before sending it to the BAM (see Fig. 28b for an example). This would allow for time series representing different goals to be stored in the same BAM without the need for relearning.

The next steps are to implement delayed reward like in the original Q-Learning task to compare the results with the original Q-Learning study as a benchmark (Teknomo, 2019) and to implement the E-E trade-off with different strategies. The challenge of how we decide what is important enough to be encoded into memory remains. Another challenge with the current model is that we need to reset the selection parameters between each full trial (after exiting the maze). The one-to-many problem can also be investigated further, rather than just selecting a “correct”

pathway at the start. Finally, the present model uses instant reward, but future studies could look into implementing delayed reward, where the reward is only given after the network has exited the maze completely, and individual actions are not rewarded or punished.

Figure 27

Example of some pathways taken starting in room A

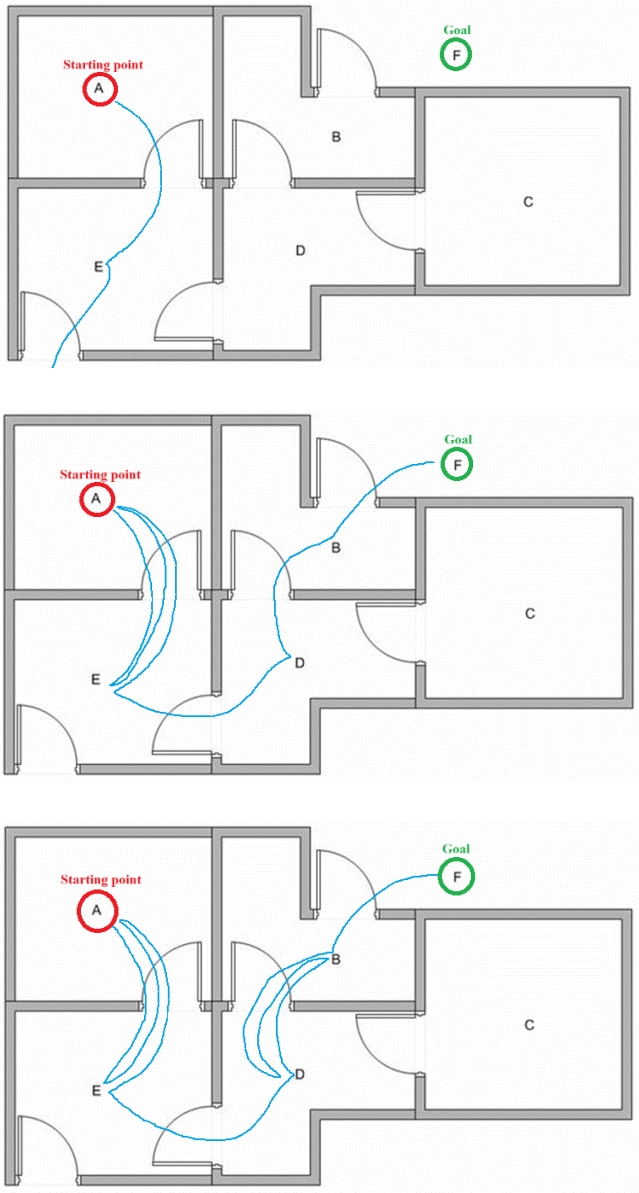
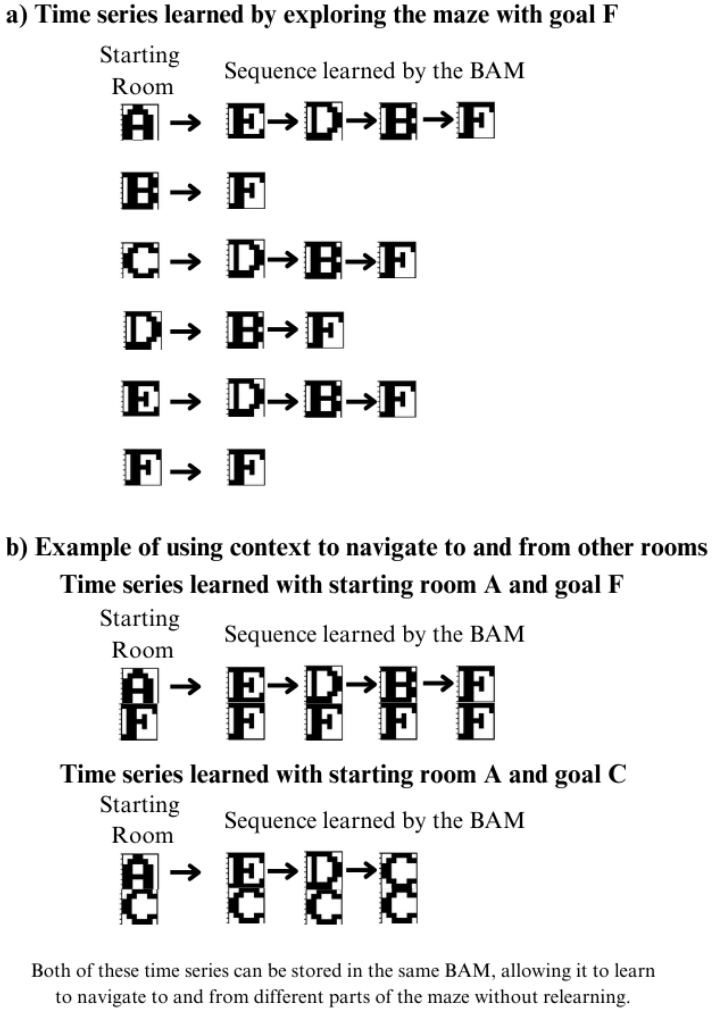


Figure 28

Time series learned by the BAM after exploring the maze



Modelling the trade-off

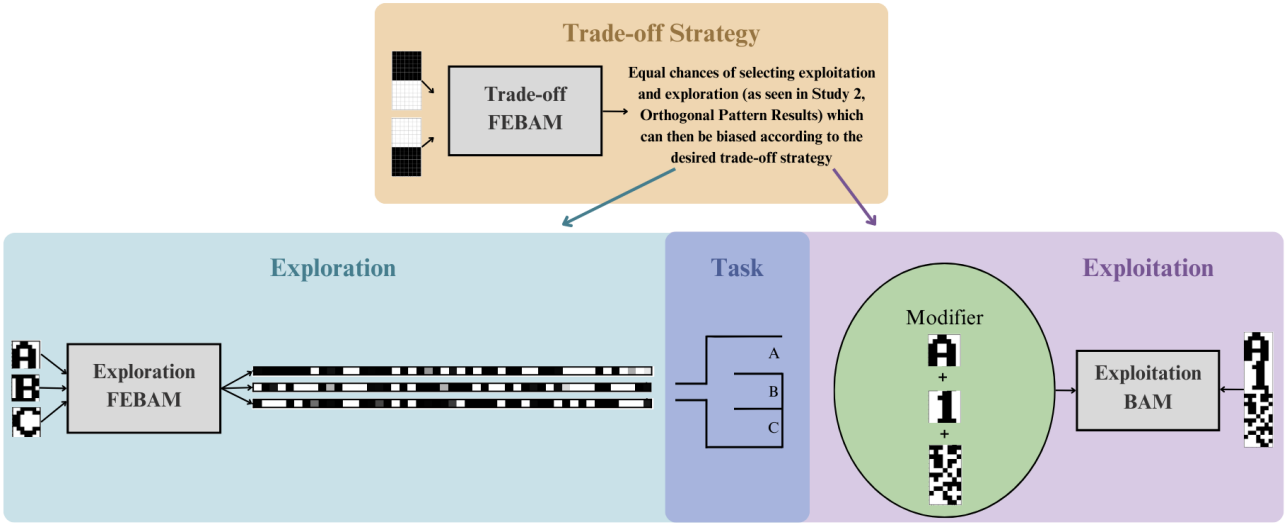
The last remaining piece left to modelling the trade-off is to combine the two studies in order to model trade-off strategies. These different trade-off strategies between the two networks can range anywhere from pure exploration (Study 2) to pure exploitation (Study 1) and everything in between. One well-known trade-off strategy, especially in Q-Learning, is the ϵ-

greedy strategy (Sutton & Barto, 1999, p. 127). The ϵ-greedy strategy is when exploitation and exploration are balanced by randomly exploring a certain percentage of the time. This could be modelled by adding a modified version of the FEBAM from Study 2 to generate randomness as needed (see Fig. 29) and to serve as the likelihood function (the percentage of time that the model will explore). It could be biased in the same way that random decisions were biased in Study 2.

For example, to get a base trade-off strategy where the network selects exploitation 50% of the time and exploration 50% of the time, one would just need to send two orthogonal inputs to the FEBAM as input to represent exploitation and exploration. As seen in Study 2, this will recall each option randomly, approximately 50% of the time (see Fig. 15a).

Figure 29

Proposed Flowchart for Combining the Two Previous Studies, with an Added FEBAM Before Every Decision to Represent an ϵ-greedy Trade-off Strategy



If one wanted to implement an ϵ-greedy strategy where it continues to explore 10% of the time, even when it has reached the optimal solution, they could send 10 different orthogonal patterns as input to the FEBAM and assign one of them to represent exploration and the rest to represent exploitation. This will result in the FEBAM recalling exploration approximately 10% of the time, as seen in Study 2 with the orthogonal distribution (see Fig. 15b). This will result in a model that exploits 90% of the time (choosing the optimal solution) and explores 10% of the time (testing to see if it can learn any new information about the other options). Exploring randomly 10% of the time is a good way to make sure it has not fallen into a local minimum or that it is not missing out on better options (Gomes & Kowalczyk, 2009; Wunder, Littman & Babes, 2010).

This representation of the trade-off could then be modified to represent any of the other various trade-off strategies found in human behaviour. Previous experiments show that children explore more when they are younger and exploit more as they age (Blanco & Sloutsky, 2020; Schulz et al., 2019), which could be accomplished by decreasing the likelihood function over time. Other studies show that exploration is only necessary when the success rate is low (Teng, Tan & Tan, 2012), so another trade-off strategy could also be based on performance or error rate. One example of an influence on our trade-off strategies is our emotional state. Previous work in cognition has shown that we are more likely to explore when we are experiencing positive emotions and we are more likely to stick to exploitation when we are experiencing fear or negative emotions (Parussel & Cañamero, 2007). This could be modelled by adding a unit to represent affect that influences the rate of exploration.

Another factor is our perceived time horizon, or how much time we believe is left to complete the given task. If we think that there is lots of time remaining, we are more likely to

explore more creating solutions. If we feel the pressure of the time constraints, we are more likely to stick to exploitation (Carstensen et al., 1999). Other studies show that the balance between the two strategies relies on our perception of environmental change (Ishii et al., 2002), how we optimize for different goals (Dimitrakakis, 2006; Hwang et al., 2003) and different tasks (Dimitrakakis, 2006). Finally, individual differences or cues that indicate a potential payoff play a huge role in which strategy we favour (Reader, 2015). All of the above can be represented by modifying the likelihood of selecting exploration vs. exploitation depending on the task or simulation desired. This biased randomness added to the trade-off strategy will allow the many of the different trade-off strategies seen in real life to be used.

Applying the model based on Q-Learning

There are few studies that propose an architecture based on artificial neural networks to implement Q-Learning (Hwang et al., 2003; Parussel & Cañamero, 2007). The models that do exist are focused on optimal responses rather than the underlying cognitive mechanisms behind decision making. In addition, the agents in the models are either unable to interact with their external environment or do not function in a dynamic environment. This is a crucial aspect of learning that should be taken into account, because humans and animals do not learn in a void. By applying the first two studies, creating a neuronal implementation of Q-Learning will be possible, improving on the models that currently exist. One could then study how different factors like previous experience and reinforcement influence the way Q-Learning works. The present two studies can be combined with a modified version of the random FEBAM to solve different versions of the maze task seen in RL with a similar level of performance as seen in classic Q-Learning. It also has additional properties that makes it more interesting from a

cognitive perspective. For example, the reward and randomness are represented in a distributed fashion. It also takes fewer trials to learn how to escape the maze.

One new method tested for representing reward was Active Unit Based Reward (AUBR), where higher levels of reward are represented by a larger number of active units (1s, represented by black pixels) and lower levels of reward are represented by the more inactive units (0s, represented by white pixels). This representation was inspired by how the spiking frequency of neurons is related to the level of reinforcement (Cooper, 2002; Freed, 2022). In a preliminary study, it was shown that by using AUBR in a recurrent neural network, the network was able to recall stimuli based on a given or approximate level of reward, and vice versa. It was able to recall similarly rewarded patterns and give the highest and lowest rewarded patterns even when the maximum and minimum levels of reward were not used in training (Church, Bolic & Chartier, 2022).

By using different levels of reward for recall (not just MaxQ like in classical Q-Learning), alternative learned options can be recalled by the network (for example, the pattern with moderate levels of reward or the low levels of reward can be recalled in addition to the highest rewarded option if AUBR is used, whereas in Q-Learning only the highest rewarded option is recalled). Also, unlike Q-Learning models, it will not be limited to a specific task. As seen in Study 1, many different series of responses can be learned by the same network when context is used to distinguish between the different tasks.

Study Limitations

One issue with the current implementation is that the model of exploitation is quite limited. It currently represents pure exploitation, meaning that it is only iterating through known responses and stabilizing on the correct answer. There is no mechanism in place for altering the

order of learned responses for a given task based on new information. Future studies would need to find a way of representing the whole spectrum between pure exploration and pure exploitation, including the challenge of how some things can be unlearned or the order of our responses can be altered as we gather new information. Another consideration for future applied studies is the temporal aspect. By accounting for *when* decisions are made and *when* actions should be accomplished, it would ease the transition from numerical simulation to real-time neurorobotic implementation.

Finally, the use of the FEBAM in Study 1 to generate unique representations may not be optimal, given recent research that has shown better performance from FEBAMs with multiple layers combined with a BAM (MF-BAM) on non-linearly separable tasks (D. Rolon-Mérette, T. Rolon-Mérette & Chartier, 2023). Future research could explore applying this new model to the E-E dilemma and the implementations above.

Conclusion

The goal for this thesis was to create a neural implementation of pure exploration and pure exploitation using recurrent neural networks, with a dynamical neural architecture. Study 1 presents an implementation of pure exploitation, using environmental feedback and context to accomplish many tasks in the same network. Study 2 presents an implementation of pure exploration including randomness, behaviour generation, and reward. This implementation offers new properties like the distribution of information across a network instead of localized in a single neuron or stored as a single value. Potential future work is discussed, including how to combine the two studies to model the E-E trade-off and how to apply these models to other tasks in cognition like the maze task. Being able to model both exploration and exploitation offers new understanding into our own cognition and how we learn from the dynamic world around us.

References

Akanksha, E., Sharma, N., & Gulati, K. (2021, April). Review on reinforcement learning, research evolution and scope of application. In *2021 5th international conference on computing methodologies and communication (ICCMC)*, 1416-1423. https://doi.org/10.1109/ICCMC51019.2021.9418283

Almardeny, Y., Benavoli, A., Boujnah, N., & Naredo, E. (2022). A reinforcement learning system for generating instantaneous quality random sequences. *IEEE Transactions on Artificial Intelligence.* https://doi.org/10.1109/TAI.2022.3161893.

Audibert, J. Y., Munos, R., & Szepesvari, C. (2009). Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science, 410*(19), 1876-1902.

Ayton, P., Hunt, A. J., & Wright, G. (1989). Psychological conceptions of randomness. *Journal of Behavioral Decision Making, 2*(4), 221–238. https://doi.org/10.1002/bdm.3960020403.

Bégin, J., & Proulx, R. (1996). Categorization in unsupervised neural networks: The Eidos model. *IEEE Transactions on Neural Networks*, *7*(1), 147–154. https://doi.org/10.1109/72.478399

Berridge, K. C., & Robinson, T. E. (2003). Parsing reward. *Trends in Neuroscience, 26,* 507-512. https://doi.org/10.1016/S0166-2236(03)00233-9.

Besbes, O., Gur, Y., & Zeevi, A. (2019). Optimal Exploration–Exploitation in a Multi-armed Bandit Problem with Non-stationary Rewards. *Stochastic Systems*, *9*(4), 319–337. https://doi.org/10.1287/stsy.2019.0033

Bhandari, A., & Duncan, J. (2014). Goal neglect and knowledge chunking in the construction of novel behaviour. *Cognition*, *130*(1), 11. https://doi.org/10.1016/J.COGNITION.2013.08.013

Blanchard, T. C., & Gershman, S. J. (2018). Pure correlates of exploration and exploitation in the human brain. *Cognitive, Affective, & Behavioral Neuroscience*, *18*(1), 117–126. https://doi.org/10.3758/s13415-017-0556-2

Blanco, N. J., & Sloutsky, V. M. (2020). Systematic exploration and uncertainty dominate young children's choices. *Developmental Science*. https://doi.org/10.1111/desc.13026

Blum, L., Shub, M., & Blum, M. (1986). A simple unpredictable pseudo-random number generator. *SIAM Journal on computing, 15*(2), 364-383. https://doi.org/10.1137/0215025.

Brown, V. M., Hallquist, M. N., Frank, M. J., & Dombrovski, A. Y. (2022). Humans adaptively resolve the explore-exploit dilemma under cognitive constraints: Evidence from a multi-armed bandit task. *Cognition, 229*(2022): 105233.

Carstensen, L. L., Isaacowitz, D. M., & Charles, S. T. (1999). Taking time seriously: A theory of socioemotional selectivity. *American Psychologist*, *54*(3), 165–181. https://doi.org/10.1037/0003-066X.54.3.165

Chapelle, O., & Li, L. (2011). An empirical evaluation of Thompson sampling. *Advances in neural information processing systems, 24.*

Chartier, S., & Boukadoum, M. (2006). A Sequential Dynamic Heteroassociative Memory for Multistep Pattern Recognition and One-to-Many Association. *IEEE Transactions on Neural Networks*, *17*(1), 59–68. https://doi.org/10.1109/TNN.2005.860855

Chartier, S., Boukadoum, M., & Amiri, M. (2009). BAM learning of nonlinearly separable tasks by using an asymmetrical output function and reinforcement learning. *IEEE Transactions on Neural Networks*, *20*(8), 1281–1292. https://doi.org/10.1109/TNN.2009.2023120

Chartier, S., Giguere, G., Renaud, P., Lina, J.-M., & Proulx, R. (2007). FEBAM: A feature-extracting bidirectional associative memory. *2007 International Joint Conference on Neural Networks*, Orlando, FL, USA, 1679-1684. https://doi.org/10.1109/IJCNN.2007.4371210.

Chen, Y. P., Nelson, L. D., & Hsu, M. (2015). From "Where" to "What": Distributed Representations of Brand Associations in the Human Brain. *52*(4), 453–466. https://doi.org/10.1509/JMR.14.0606

Church, K., Bolic, M., & Chartier, S. (2022). La représentation de la récompense au sein d'un réseau de neurones artificiels. In *44e congrès annuel de la Société Québécoise pour la Recherche en Psychologie.*

Church, K.A., Ross, M., & Chartier, S. (2020, July). Using a Bidirectional Associative Memory and Feature Extraction to model Nonlinear Exploitation Problems. In Steward, T. C. (Ed.). *Proceedings of the 19th International Conference on Cognitive Modelling* (pp. 37-42). University Park, PA: Applied Cognitive Science Lab, Penn State.

Cohen, J., Cohen, P., West, S., & Aiken, L. (2013). *Applied multiple regression/correlation analysis for the behavioral sciences*. Routledge. https://doi.org/10.4324/9780203774441.

Cohen, J. D., McClure, S. M., & Yu, A. J. (2007). Should I stay or should I go? How the human brain manages the trade-off between exploitation and exploration. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *362*(1481), 933–942. https://doi.org/10.1098/rstb.2007.2098

Cooper, D. C. (2002). The significance of action potential bursting in the brain reward circuit. *Neurochemistry International*, *41*(5), 333–340. https://doi.org/10.1016/S0197-0186(02)00068-2

Croson, R., & Sundali, J. (2005). The Gambler's Fallacy and the hot hand: Empirical data from casinos. *Journal of Risk and Uncertainty*, *30*(3), 195–209. https://doi.org/10.1007/s11166-005-1153-2.

Daw, N. D., O'Doherty, J. P., Dayan, P., Seymour, B., & Dolan, R. J. (2006). Cortical substrates for exploratory decisions in humans. *Nature*, *441*(7095), 876–879. https://doi.org/10.1038/nature04766

De Bernardi, M., Khouzani, M. H. R., & Malacaria, P. (2019). Pseudo-Random Number Generation Using Generative Adversarial Networks. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 18*, 191-200. https://doi.org/10.1007/978-3-030-13453-2_15.

Del Giudice, M., & Crespi, B. J. (2018). Basic functional trade-offs in cognition: An integrative framework. *Cognition*, *179*, 56–70. https://doi.org/10.1016/j.cognition.2018.06.008

Desai, V., Ravindra, P., & Dandina, R. (2012). Using layer recurrent neural network to generate pseudo random number sequences. *International Journal of Computer Science Issues, 9*(2), 324-334.

Dimitrakakis, C. (2006). Nearly optimal exploration-exploitation decision thresholds. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4131 LNCS-I, 850–859. https://doi.org/10.1007/11840817_88

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, *14*(2), 179–211. https://doi.org/10.1207/s15516709cog1402_1

Elyada, Y. M., & Horn, D. (2005, September). Can dynamic neural filters produce pseudo-random sequences? In *Artificial Neural Networks: Biological Inspirations-ICANN 2005: 15th International Conference,* Warsaw, Poland (pp. 211-216). Springer Berlin Heidelberg. https://doi.org/10.1007/11550822_34.

Ferreira, K. J., Simchi-Levi, D., & Wang, H. (2018) Online network revenue management using Thompson sampling. *Operations research, 66*(6), 1586-1602.

Freed, W. J. (2022). How the Brain Signals Reward. In *Motivation and Desire* (pp. 129–137). Springer International Publishing. https://doi.org/10.1007/978-3-031-10477-0_13

Gershman, S. J., & Niv, Y. (2015). Novelty and Inductive Generalization in Human Reinforcement Learning. *Topics in Cognitive Science*, *7*(3), 391–415. https://doi.org/10.1111/tops.12138

Gobet, F., Lane, P. C. R., Croker, S., Cheng, P. C. H., Jones, G., Oliver, I., & Pine, J. M. (2001). Chunking mechanisms in human learning. *Trends in Cognitive Sciences*, *5*(6), 236–243. https://doi.org/10.1016/S1364-6613(00)01662-4

Gomes, E. R., & Kowalczyk, R. (2009). Dynamic analysis of multiagent Q-learning with ϵ-greedy exploration. *ACM International Conference Proceeding Series*, *382*. https://doi.org/10.1145/1553374.1553422

Guseva, M., Bogler, C., Allefeld, C., & Haynes, J. D. (2023). Instruction effects on randomness in sequence generation. *Frontiers in Psychology, 14,* 1113654. https://doi.org/10.3389/fpsyg.2023.1113654

Hallquist, M. N., & Dombrovski, A. Y. (2019). Selective maintenance of value information helps resolve the exploration/exploitation dilemma. *Cognition*, *183*, 226–243. https://doi.org/10.1016/j.cognition.2018.11.004

Hélie, S., & Sun, R. (2010). Incubation, insight, and creative problem solving: a unified theory and a connectionist model. *Psychological Review*, *117*(3), 994. https://doi.apa.org/doiLanding?doi=10.1037%2Fa0019532

Héricé, C., Khalil, R., Moftah, M., Boraud, T., Guthrie, M., & Garenne, A. (2016). Decision making under uncertainty in a spiking neural network model of the basal ganglia. *Journal of Integrative Neuroscience*, *15*(04), 515–538. https://doi.org/10.1142/S021963521650028X

Hills, T. T., Todd, P. M., Lazer, D., Redish, A. D., & Couzin, I. D. (2015). Exploration versus exploitation in space, mind, and society. *Trends in Cognitive Sciences*, *19*(1), 46–54. https://doi.org/10.1016/J.TICS.2014.10.004

Hughes, J. M. (2007). *Pseudo-random Number Generation Using Binary Recurrent Neural Networks* Doctoral dissertation, Kalamazoo College.

Hwang, K. S., Chiou, J. Y., & Wu, C. S. (2003). A neural network solution for exploitation and exploration problems. *Proceedings of 2003 International Conference on Neural Networks and Signal Processing, ICNNSP'03*, *1*, 54–57, Vol. 1. https://doi.org/10.1109/ICNNSP.2003.1279211

Ishii, S., Yoshida, W., & Yoshimoto, J. (2002). Control of exploitation-exploration meta-parameter in reinforcement learning. *Neural Networks*, *15*(4–6), 665–687. https://doi.org/10.1016/S0893-6080(02)00056-4

Jeong, Y. S., Oh, K., Cho, C. K., & Choi, H. J. (2018, January). Pseudo random number generation using LSTMs and irrational numbers. In *2018 IEEE international conference on big data and smart computing (BigComp)* (pp. 541-544). IEEE. https://doi.org/10.1109/BigComp.2018.00091.

Jokar, E., & Mikaili, M. (2012). Assessment of Human Random Number Generation for Biometric Verification. *Journal of Medical Signals and Sensors, 2*(2), 82–87.

Jordan, M. I. (1997). Serial order: A parallel distributed processing approach. In *Advances in Psychology* (Vol. 121, Issue C, pp. 471–495). Elsevier. https://doi.org/10.1016/S0166-4115(97)80111-2

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, *4*, 237–285. https://doi.org/10.1613/jair.301

Kanerva, P. (2009). Hyperdimensional Computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation, 1*(2), 139–159. https://doi.org/10.1007/s12559-009-9009-8.

Khalil, R., & Moustafa, A. A. (2022). A neurocomputational model of creative processes. *Neuroscience & Biobehavioral Reviews, 137,* 104656. https://doi.org/10.1016/j.neubiorev.2022.104656.

Laureiro-Martínez, D., Canessa, N., Brusoni, S., Zollo, M., Hare, T., Alemanno, F., & Cappa, S. F. (2014). Frontopolar cortex and decision-making efficiency: comparing brain activity of experts with different professional background during an exploration-exploitation task. *Frontiers in Human Neuroscience*, *7*, 927. https://doi.org/10.3389/fnhum.2013.00927

Lew, S., Rey, H. G., & Zanutto, B. S. (2013). Neuronal mechanisms underlying exploration-exploitation strategies in operant learning. *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 1–6. https://doi.org/10.1109/IJCNN.2013.6706987

Lopez-Persem, A., Domenech, P., & Pessiglione, M. (2016). How prior preferences determine decision-making frames and biases in the human brain. *ELife, 5*, e20317. https://doi.org/10.7554/eLife.20317.

Malik, K., Pulikkotil, J., & Sharma, A. (2021). Comparison of pseudorandom number generators and their application for uncertainty estimation using Monte Carlo Simulation. *MAPAN*, *36*(3), 481–496. https://doi.org/10.1007/s12647-021-00443-3.

Man, Z., Li, J., Di, Z., Liu, X., Zhou, J., Wang, J., & Zhang, X. (2021). A novel image encryption algorithm based on least squares generative adversarial network random number generator. *Multimedia Tools and Applications, 80,* 27445-27469. https://doi.org/10.1007/s11042-021-10979-w.

Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS), 8*(1), 3-30. https://doi.org/10.1145/272991.272995.

Mehlhorn, K., Newell, B. R., Todd, P. M., Lee, M. D., Morgan, K., Braithwaite, V. A., Hausmann, D., Fiedler, K., & Gonzalez, C. (2015). Unpacking the exploration-exploitation tradeoff: A synthesis of human and animal literatures. *Decision*, *2*(3), 191–215. https://doi.org/10.1037/dec0000033

Monk, C. T., Barbier, M., Romanczuk, P., Watson, J. R., Alós, J., Nakayama, S., Rubenstein, D. I., Levin, S. A., & Arlinghaus, R. (2018). How ecology shapes exploitation: a framework to predict the behavioural response of human and animal foragers along exploration-exploitation trade-offs. *Ecology Letters*, *21*(6), 779–793. https://doi.org/10.1111/ele.12949

Naruse, M., Yamamoto, E., Nakao, T., Akimoto, T., Saigo, H., Okamura, K., Ojima, I., Northoff, G., & Hori, H. (2018). Local reservoir model for choice-based learning. *PLoS ONE*, *13*(10). http://arxiv.org/abs/1804.04324

Nickerson, R. S. (2002). The production and perception of randomness. *Psychological Review, 109*(2), 330–357. https://doi.org/10.1037/0033-295X.109.2.330.

Orquin, J. L., & Loose, S. M. (2013). Attention and choice: A review on eye movements in decision making. *Acta psychologica, 144*(1), 190-206. https://doi.org/10.1016/j.actpsy.2013.06.003.

O'Toole, A. J., Jiang, F., Abdi, H., & Haxby, J. V. (2005). Partially Distributed Representations of Objects and Faces in Ventral Temporal Cortex. *Journal of Cognitive Neuroscience*, *17*(4), 580–590. https://doi.org/10.1162/0898929053467550

Park, J., Hong, J. P., Kim, H., & Jeong, B. J. (2023). Auto-Encoder Based Orthogonal Time Frequency Space Modulation and Detection with Meta-Learning. *IEEE Access, 11,* 43008-43018. https://doi.org/10.1109/ACCESS.2023.3271993.

Park, S., Kim, K., Kim, K., & Nam, C. (2022). Dynamical pseudo-random number generator using reinforcement learning. *Applied Sciences, 12*(7), 3377. https://doi.org/10.3390/app12073377.

Parussel, K., & Cañamero, L. (2007). *Biasing Neural Networks Towards Exploration or Exploitation Using Neuromodulation*, 889–898. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-74695-9_91

Pasqualini, L., & Parton, M. (2020). Pseudo random number generation: A reinforcement learning approach. *Procedia Computer Science, 170,* 1122-1127. https://doi.org/10.1016/j.procs.2020.03.057.

Reader, S. M. (2015). Causes of Individual Differences in Animal Exploration and Search. *Topics in Cognitive Science*, *7*(3), 451–468. https://doi.org/10.1111/tops.12148

Rissman, J., & Wagner, A. D. (2012). Distributed representations in memory: Insights from functional brain imaging. *Annual Review of Psychology*, *63*, 101. https://doi.org/10.1146/ANNUREV-PSYCH-120710-100344

Rolon-Mérette, D., Rolon-Mérette, T., & Chartier, S. (2018). Learning and Recalling Arbitrary Lists of Overlapping Exemplars in a Recurrent Artificial Neural Network. *ICCM*.

Rolon-Mérette, T., Rolon-Merette, D., & Chartier, S. (2018). Generating Cognitive Context with Feature-Extracting Bidirectional Associative Memory. *Procedia Computer Science*, *145*, 428–436. https://doi.org/10.1016/j.procs.2018.11.102

Rolon-Mérette, T., Rolon-Mérette, D., & Chartier, S. (2023, May). Towards binary encoding in Bidirectional Associative Memories. *Proceedings of the Florida Artificial Intelligence Research Society Conference: FLAIRS-36*.

Rolon-Mérette, D., Rolon-Mérette, T., & Chartier, S. (2023). A multilayered bidirectional associative memory model for learning nonlinear tasks. *Neural Networks, 167,* 244-265. https://doi.org/10.1016/j.neunet.2023.08.018

Schulz, E., Wu, C. M., Ruggeri, A., & Meder, B. (2019). Searching for Rewards Like a Child Means Less Generalization and More Directed Exploration. *Psychological Science*, *30*(11), 1561–1572. https://doi.org/10.1177/0956797619863663

Small, S. L., Hart, J., Nguyen, T., & Gordon, B. (1995). Distributed representations of semantic knowledge in the brain. *Brain*, *118*(2), 441–453. https://doi.org/10.1093/BRAIN/118.2.441

Stafford, T., Thirkettle, M., Walton, T., Vautrelle, N., Hetherington, L., Port, M., Gurney, K., & Redgrave, P. (2012). A Novel Task for the Investigation of Action Acquisition. *PLoS ONE*, *7*(6), e37749. https://doi.org/10.1371/journal.pone.0037749

Storkey, A. J., & Valabregue, R. (1999). The basins of attraction of a new Hopfield learning rule. *Neural Networks*, *12*(6), 869–876. https://doi.org/10.1016/S0893-6080(99)00038-6

Sutton, R. S., & Barto, A. G. (1999). Reinforcement Learning. *Journal of Cognitive Neuroscience* , *11*(1), 126–134.

Sutton, R. S., & Barto, A. G. (2017). *Reinforcement Learning: An Introduction* (2nd ed.). The MIT Press.

Teknomo, K. (2019). *Q-Learning by examples*. https://people.revoledu.com/kardi/tutorial/ReinforcementLearning/index.html

Teng, T. H., Tan, A. H., & Tan, Y. S. (2012). Self-regulating action exploration in reinforcement learning. *Procedia Computer Science*, *13*, 18–30. https://doi.org/10.1016/j.procs.2012.09.110

Tervo, D. G. R., Proskurin, M., Manakov, M., Kabra, M., Vollmer, A., Branson, K., & Karpova, A. Y. (2014). Behavioral Variability through Stochastic Choice and Its Gating by Anterior Cingulate Cortex. *Cell*, *159*(1), 21–32. https://doi.org/10.1016/J.CELL.2014.08.037

Tilahun, S. L. (2019). Balancing the Degree of Exploration and Exploitation of Swarm Intelligence Using Parallel Computing. *International Journal on Artificial Intelligence Tools*, *28*(3). https://doi.org/10.1142/S0218213019500143

Tirdad, K., & Sadeghian, A. (2010, July). Hopfield neural networks as pseudo random number generators. In *2010 Annual meeting of the North American fuzzy information processing society* (pp. 1-6). IEEE. https://doi.org/10.1109/NAFIPS.2010.5548182.

Tremblay, C., Myers-Stewart, K., Morissette, L., & Chartier, S. (2013). Bidirectional Associative Memory and Learning of Nonlinearly Separable Tasks. In R. West & T. Stewart (Eds.), *Proceedings of the 12th International Conference on Cognitive Modeling* (pp. 420–425).

Tsuda, I. (2001). Toward an interpretation of dynamic neural activity in terms of chaotic dynamical systems. *Behavioral and Brain Sciences*, *24*(5), 793–810. https://doi.org/10.1017/s0140525x01000097

Vakili, S., Liu, K., & Zhao, Q. (2013). Deterministic sequencing of exploration and exploitation for multi-armed bandit problems. *IEEE Journal of Selected Topics in Signal Processing, 7*(5), 759-767.

Warren, P. A., Gostoli, U., Farmer, G. D., El-Deredy, W., & Hahn, U. (2018). A re-examination of "bias" in human randomness perception. *Journal of Experimental Psychology: Human Perception and Performance*, *44*(5), 663–680. https://doi.org/10.1037/xhp0000462.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. PhD thesis, King's College, Oxford.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-Learning. *Machine Learning*, *8*(3–4), 279–292.

Wittmann, B. C., Daw, N. D., Seymour, B., & Dolan, R. J. (2008). Striatal activity underlies novelty-based choice in humans. *Neuron, 58*(6), 967-973. https://doi.org/10.1016/j.neuron.2008.04.027.

Wunder, M., Littman, M., & Babes, M. (2010). Classes of Multiagent Q-learning Dynamics with-greedy Exploration. *ICML*, 1167–1174.