Generating Distributed Randomness using Artificial Neural Networks

Kinsey Antonina Church (kchur026@uottawa.ca)
School of Psychology, 136 Jean-Jacques Lussier, Vanier Hall
Ottawa, ON, K1N 6N5, CAN

Marija Bolic (mboli027@uottawa.ca)
School of Psychology, 136 Jean-Jacques Lussier, Vanier Hall
Ottawa, ON, K1N 6N5, CAN

Sylvain Chartier (sylvain.chartier@uottawa.ca)
School of Psychology, 136 Jean-Jacques Lussier, Vanier Hall
Ottawa, ON, K1N 6N5, CAN

Abstract

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

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

In L. K. Samuelson, S. L. Frank, M. Toneva, A. Mackey, & E. Hazeltine (Eds.), *Proceedings of the 46th Annual Conference of the Cognitive Science Society.*

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; Rolon-Mérette, 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 $\mathbf{x}_{[0]}$and $\mathbf{y}_{[0]}$ are the initial inputs (Fig. 1). The weight matrices, **W** and **V**, connect the network units, **x** and **y**, and return information to each other. The inputs $\mathbf{x}_{[c]}$ and $\mathbf{y}_{[c]}$ are the inputs at the current iteration number *c*.

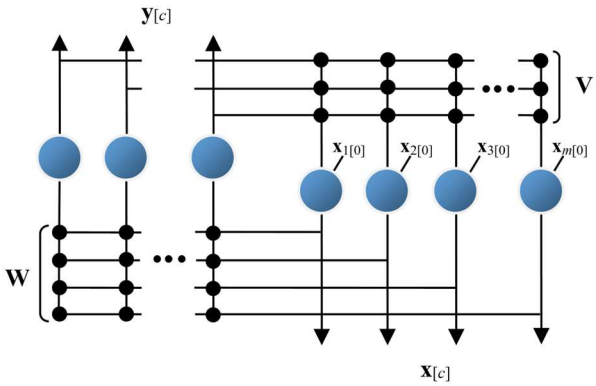

Figure 1: FEBAM Architecture.

Transmission

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

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

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 $\mathbf{Wx}_{i[c]}$ with $\mathbf{Vy}_{i[c]}$.

Learning

The learning is based on time difference Hebbian learning:

$$\mathbf{W}_{[k+1]} = \mathbf{W}_{[k]} + \eta\left(\mathbf{y}_{[0]} - \mathbf{y}_{[c]}\right)\left(\mathbf{x}_{[0]} + \mathbf{x}_{[c]}\right)^{\mathrm{T}} \quad (2)$$

Where η is a small positive learning parameter, **W** is the weight matrix, $\mathbf{x}_{[0]}$ is the initial pattern, and $\mathbf{y}_{[0]}$ is the first generated representation; **V** is updated using an equivalent function. For learning, the cycle index, *c*, is set to 1 throughout the simulations.

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 2). 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.

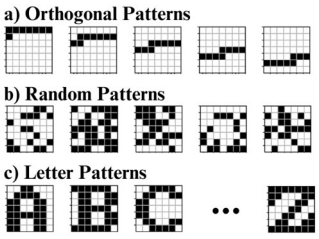

Figure 2: Examples of different input patterns of 49 dimensions.

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}.

Learning Procedure The learning was accomplished as follows:
1. Selection of input patterns (Fig. 2)
2. Computation of the output (Eq. 1) and update of the weights (Eq. 2).
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), 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 2a 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 3, 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. 3a) or the multinomial one, $n = 5$, χ^2 (1, 199) = 2.19, $p = 0.7$ (Fig. 3b).

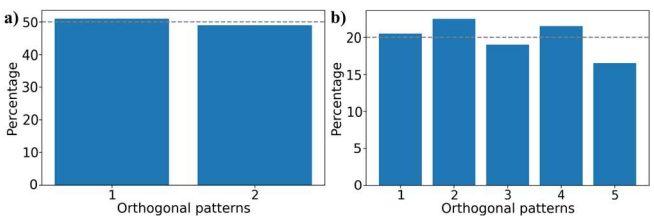

Figure 3: Binomial and multinomial distribution of orthogonal patterns, with the expected percentage.

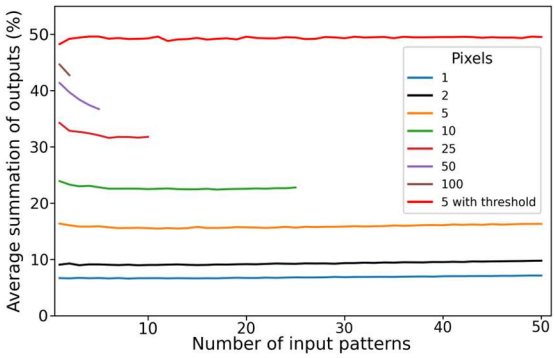

Figure 4: Orthogonal patterns with different pixels in input patterns.

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. 4). 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.

Random and Structured Correlated Patterns

In a more natural setting, input patterns are correlated; whether randomly generated (Fig. 2b) or structured (like uppercase letters; Fig. 2c). 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 5. 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. 6). 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.

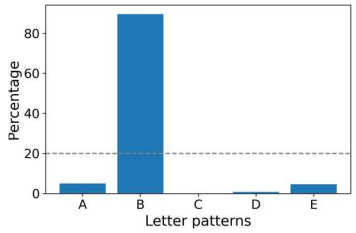

Figure 5: Distribution of letter patterns, with the expected percentage.

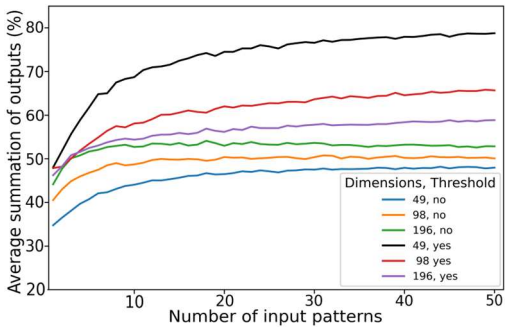

Figure 6: Random inputs with and without the threshold function applied, 3 different dimensions.

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 3b).

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 to 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 5, 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. 2c) 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:

$$\mathbf{o}_i = \boldsymbol{\beta}_i \sum_j \mathbf{y}_{ij} \tag{3a}$$

$$\mathrm{s} = \mathrm{Max}(\mathbf{o}_i) \tag{3b}$$

Where $\mathbf{y}_i$ represents the output pattern i, $\boldsymbol{\beta}$, the selection parameter, j the given pixel (j…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 value closer to 0 for slow transition:

$$\boldsymbol{\beta}_{i[t+1]} = \begin{cases} 0, \ if \ (\boldsymbol{\beta}_{i[t]} - \varepsilon) < 0 \\ \boldsymbol{\beta}_{i[t+1]}, Else \end{cases} \tag{4}$$

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. 7, "Incorrect Response"). When the correct output is selected, then the selection parameter remains the same (see Fig. 7, "Correct Response").

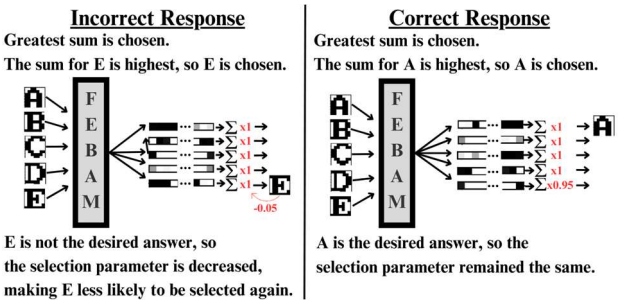

Figure 7: Flowchart for the Desired Output Condition.

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. 2c). 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. 1).
3. Each representation is then modified using Eq. 3 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. 4.
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. 8a), it takes almost 11 trials for selecting A. Conversely, when the selection parameter is decreased with greater quantity (0.5; Fig. 8b) 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 9.

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 10a). 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. 10b).

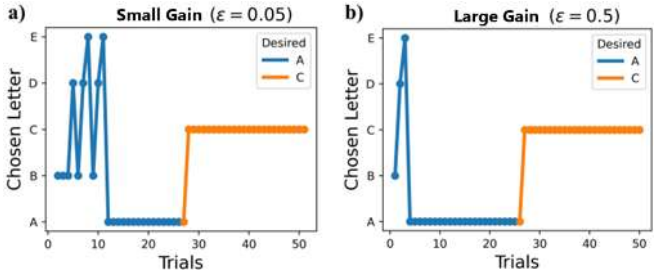

Figure 8: Difference in slow vs. fast selection parameter for desired letter A for 25 trials and C for 25 trials.

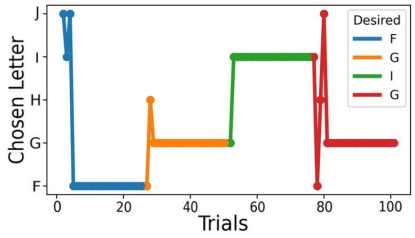

Figure 9: Chosen letter over 100 trials, where the desired letter is F for 25 trials, then G, I, and G again.

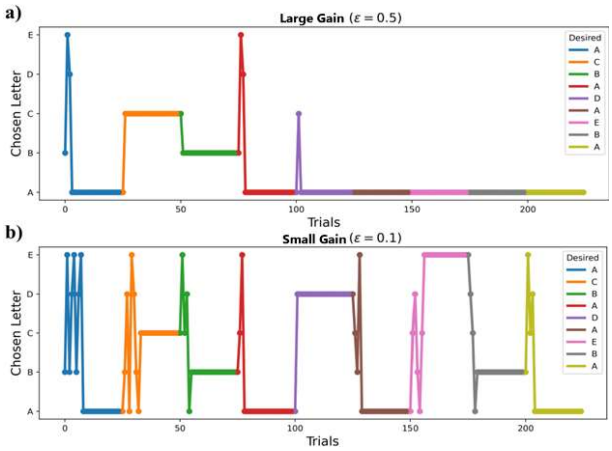

Figure 10: Chosen letter over 250 trials where the desired letter is changing every 25 trials.

Sampling without Replacement Condition

For this condition, a series of orthogonal or correlated letter patterns are learned by the FEBAM (Fig. 2a and 2c). The goal

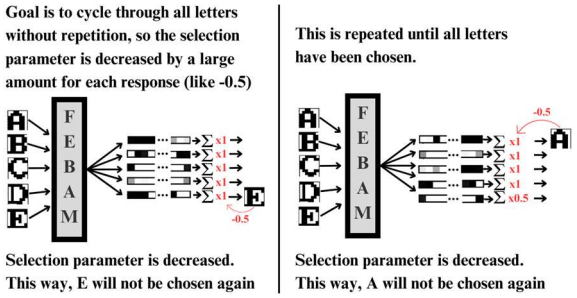

Figure 11: Flowchart for the Sampling without Replacement Condition.

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. 11).

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. 12) 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. 12), 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, $p < 0.01$).

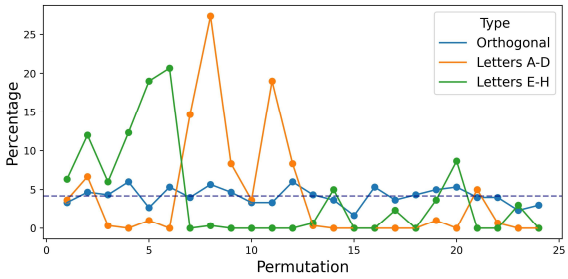

Figure 12: Orthogonal and Letter sampling without replacement trials with theoretical frequency.

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 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.

Acknowledgements

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

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.

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.

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.

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.

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.

Church, K., Ross, M., & Chartier, S. (2020). Using a Bidirectional Associative Memory and Feature Extraction to model Nonlinear Exploitation Problems. *Proceedings of the International Conference on Cognitive Modelling*, 37-43.

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.

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.

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.

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.

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.

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

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

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.

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.

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.

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.

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.

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.

Ranasinghe, K., Naseer, M., Hayat, M., Khan, S., & Khan, F. S. (2021). Orthogonal Projection Loss. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 12333-12343).

Rolon-Mérette, T., Rolon-Mérette, 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. *The International FLAIRS Conference Proceedings, 36*(1). https://doi.org/10.32473/flairs.36.133365.

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.

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.

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.