Using Bidirectional Associative Memory Neural Networks to Solve the *N*-bit Task

Damiem Rolon-Mérette[1a], Thaddé Rolon-Mérette[1b], & Sylvain Chartier[1c]

[1]School of Psychology at the University of Ottawa

[a]Drolo083@uottawa.ca, [b]Trolo068@uottawa.ca , [c]Sylvain.chartier@uottawa.ca

Abstract

Nowadays, artificial neural networks can easily solve the *N*-bit parity problem. However, each time a different level must be learned, the network must be retrained. This, combined with the exponential increase of learning trials required as *N* grows, make these models too different from how their biological counterpart solves them. This is because humans learn to recognize patterns, count, and determine if numbers are odd or even. Once they have learned these tasks, they can have them interact to solve any level without further training. This behavior is akin to performing multiple associations of different tasks. Therefore, it is proposed that by using bidirectional associative memory neural networks, it would be possible to solve the *N*-bit parity problem in a similar fashion to humans. To achieve this, two networks interacted; one served as a task Identifier and the other as a memory Extractor, giving the desired behavior influenced by the Identifier. Results showed that the model could solve the 2- to 9-bit in linear time once the associations were learned. Moreover, this was possible with 97% fewer inputs and no retraining. In addition, because of the recurrent nature of the model, it could also solve the tasks even under high noise levels.

Introduction

In the last decade, artificial neural networks (ANNs) have grown in popularity, largely due to their capability to tackle complex tasks [Cichy & Kaiser, 2019; Silver et al., 2016]. That being said, when drawing the parallel between ANNs and human cognition [Anderson et al., 1977; Martindale, 1991; Wang & Cui, 2018], there are concerns about how they achieve this [Marcus, 2018]. This becomes evident when revisiting the classic benchmark, the XOR [Marvin & Seymour, 1969], and its generalization, the *N*-bit parity task.

For an ANN, these input-target pairing tasks require learning to associate different input vectors of binary values (1 or 0) of length *N*, to their corresponding targets (1 or 0), through rote associations. Nowadays, solving the *N*-bit parity task has become trivial, as many feedforward neural networks can do it with ease [Dhar et al., 2010; Yanling, Bimin & Zhanrong, 2002; Yang, Yang & Wu, 2011]. However, most models must be retrained from scratch each time they have to solve a different *N*-bit, which hinders their generalizability. Also, the number of inputs a network must learn increases exponentially (2^N) with the task's level, negatively impacting learning times [Tesauro & Janssens, 1988]. Nevertheless, how ANNs are trained differs from how humans learn to solve these problems [Zador, 2019].

Evidence suggests that once a human has learned to recognize numbers, count, and determine what values are odd and even, it is sufficient to solve any *N*-bit task [Berch et al., 1999; Graham, 1999; Soto-Calvo et al. 2015]. In other words, they do not memorize each input-target pair but rather have distinct modules with different learned tasks interact to solve the problem [Barrett & Kurzban, 2006; Chandra, et al., 2018; Fang et al., 2004]. This may be one of the reasons why humans are much faster and better at generalizing than ANNs.

Interestingly, this process in humans can be seen as similar to performing multiple associations of different tasks. Thus, associative mechanisms could play a key role in solving these types of problems [Anderson & Bower, 2014; Buckner & DiNicola, 2019; Darcey, 2016, 2020; Lansner, 2009; Mandelbaum, 2015]. As such, an interesting avenue would be to use recurrent neural associative memories (RNAM) [Anderson, 1983], a type of ANN that explains how learned patterns may correspond to attractors in the brain's neuronal state space. Consequently, RNAMs find themselves in various cognitive theories [Knoblauch, 2005]. Of interest, due to its ability to learn auto- and hetero association, the bidirectional associative memory (BAM) subtype would be used to investigate if the *N*-bit can be solved using human strategies instead of pure rote associations.

Remarkably, BAMs were introduced in the 80s [Kosko, 1988] but have remained one of the most widely used types of RNAM to study associative memory. Over the years, various versions have been proposed to overcome its original

Copyright © 2023, by the authors. All rights reserved.

limitations (see [Acevedo-Mosqueda, Yanez-Marquez & Acevedo-Mosqueda, 2013] for a review). But recently, the Multiple Feature extracting Bidirectional Associative Memory (MF-BAM) has been proposed as a cognitively inspired model with interesting properties [Rolon-Mérette, Rolon-Mérette & Chartier, 2023]. This multi-network can learn any type of association while remaining plausible [O'Reilly, 1998]. It combines unsupervised and supervised learning and is based on simple local Hebbian and anti-Hebbian learning. Thus, it is an interesting candidate to use to study human cognition.

However, when learning different tasks, inputs may belong to many different sets leading to one-to-many associations [Elman, 1990; Jordan, 1997]. One remedy is using contextual information [Rolon-Mérette, Rolon-Mérette & Chartier, 2019]. Context permits a unicity representation of the inputs and makes learning multiple tasks possible [Church, Ross & Chartier, 2020].

Thus, two MF-BAMs are needed. An initial MF-BAM module will recognize incoming stimuli and generate appropriate instructions (contextual information). These would then be given to a second MF-BAM module to generate the corresponding desired behavior (tasks). In other words, by combining two MF-BAMs, it could be possible to learn all prior associations necessary to solve any *N*-bit parity problem faster and in a more general way.

The remainder of the paper divides itself as follows: Section I: Model, Section II: Methodology, Section III: Results, and Section IV: Discussion.

Section I. Model

Learning to solve the *N*-bit parity requires the interaction between two modules (MF-BAMs). The first module, the Identifier, will generate the appropriate instruction set (contextual information) from the incoming stimuli (inputs). The second module, called the Extractor, will retrieve the corresponding stored behaviors from the concatenation of the inputs and the contexts. Fig. 1a) shows the overall implementation of the proposed model. In short, an input pattern, $\mathbf{p}_{[t]}$ at time *t*, is sent to the Identifier, which will recall the according context $\mathbf{c}_{[t]}$ (instruction). This pattern will then be concatenated with a portion ($\mathbf{e}_{[t-1]}$) of the outputted behavior $\mathbf{b}_{[t-1]}$ from the Extractor from the previous time step (*t* - 1). This concatenated pattern $\mathbf{o}_{[t]} = (\mathbf{e}_{[t-1]} \circ \mathbf{c}_{[t]})$ is sent to the second module to generate the desired behavior. The Extractor's output is composed of two concatenate parts $\mathbf{b}_{[t]} = (\mathbf{z}_{[t]} \circ \mathbf{e}_{[t]})$. The output $\mathbf{e}_{[t]}$ will keep track of what number the model is currently counting, while $\mathbf{z}_{[t]}$ will give the final behavior (odd or even).

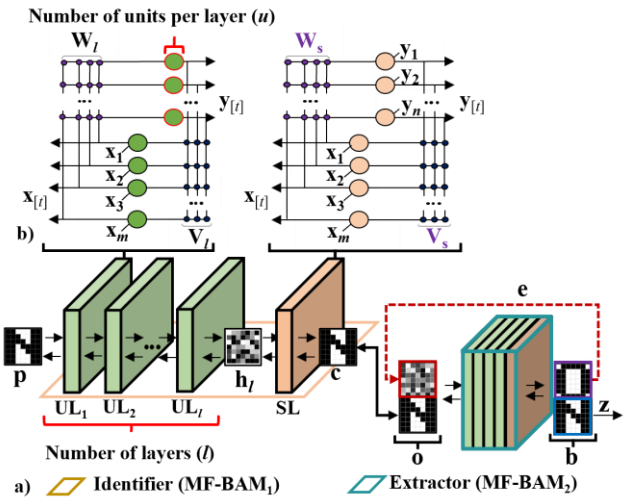

Fig. 1: a) The two MF-BAM composed of Unsupervised Layers (UL, green) and a single Supervised Layer (SL, orange), where *l* represents the number of UL and *u* the number of hidden units. b) The detailed underlying network that composes each layer.

Architecture

Each MF-BAM has two components, the MF, which includes several unsupervised networks (ULs), and the BAM, which contains a single supervised network (SL). These are all based on the unsupervised and supervised version of a modified BAM [Chartier & Boukadoum, 2006; Chartier, Giguère, Renaud, Lina & Proulx, 2007]. Fig. 1b) shows the architecture of the UL (green) and SL (orange). The MF-BAM uses the same transmission function and learning rules for all its layers. The model's behavior changes only by modifying the MF's number of y-units in each hidden layer (denoted by *u*) and the number of hidden layers (represented by *l*) used. For all units, the same cubic transmission function and for each connection Hebbian/anti-Hebbian learning rule were used. For details see [Rolon-Mérette, Rolon-Mérette & Chartier, 2023].

Section II: Methodology

Instead of learning all input-target pairs (2^N), the model will first learn to perform four different tasks selected to be sufficient to solve any parity. These tasks are:

- To recognize 1 and 0.
- To count from 0 to 9.
- To know what numbers are odd and even.
- To know when the task starts and ends.

Multiple associations will represent each task. All the simulations were performed using Python programming language installed from Anaconda [Anaconda Software Distribution, 2020; Rolon-Mérette et al., 2020] and were run on a RTX3090 graphics card.

Stimuli

For all tasks, 7x7 bipolar (values of -1 or 1) pixelated images representing alphanumerical patterns were used for the *N*-bit task and the contextual tags. These patterns represented the numbers from 0 to 9, the letters N, C, D, P, E, and O for the context *N*-bit task, Count, Fixed, and Parity, and for the targets Even and Odd, respectively. A given *N*-bit task was portrayed as a time series starting with N then, followed by the desired numbers of 0 and 1, and ending with the pattern P. Fig. 2 shows some examples of different *N*-bit.

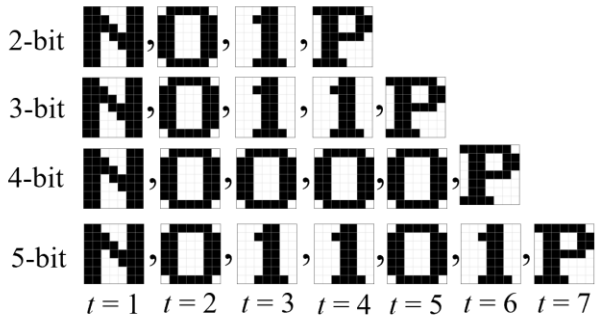

Fig. 2: Example of time series representing the input sequence during recall for 2- to 5-bit.

Input-Target Pairing

Before testing the model, each MF-BAM had to learn specific input-target pairs depicted in Fig. 3. The Identifier module had to learn a total of 4 associations directly linked to either recognizing 1 or 0 and generating the corresponding instruction (count or fixed) or recognizing to start the *N*-bit task or end it by giving the parity of the value in memory. The Extractor module had to learn 29 associations, which represented the possible actions for the numbers between 0 and 9 in the different contexts of either initializing the task, staying fixed, counting, and giving parity. Thus, a total of 33 associations were needed to solve any *N*-bit task up to 9. Also, Fig. 3 shows that for task initialization, the upper portion of the input pattern is randomly generated, ~*U*(-1,1). This was to simulate chatter noise (i.e., rest behavior).

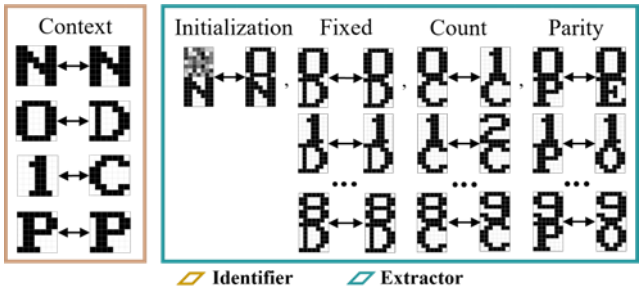

Fig. 3: Input-target pairs for each module.

Model's Flow of Information

Once the learning is accomplished, the model can be used to solve the *N*-bit parity task. The first MF-BAM receives the input stream and determines the instructions that should be sent to the second MF-BAM. As the inputs are processed, the network should be able to output the correct behavior. Fig. 4 shows this process for the 2-bit.

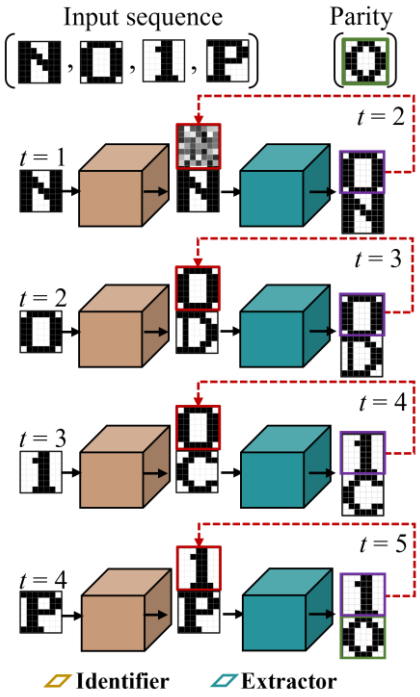

Fig. 4: Flow chart of the model when presented with the 2-bit.

Simulations Protocols

The model was tested under different conditions to better understand its capabilities.

Simulation I: The *N*-bit and Interacting Association

The model had to learn all the input-target pairings shown in Fig. 3 within 2500 epochs. This value was based on the past implementation of the MF-BAM when learning the *N*-bit [Rolon-Mérette, Rolon-Mérette & Chartier, 2023]. During recall, the entire 2- to 9-bit parity inputs were presented. Therefore, a total of 1020 ($= \sum_{i=2}^{9} 2^i$) time series were generated and presented one at a time. For reliability purposes, learning and recall were repeated 50 times. Thus, average mean squared error (MSE), performances, and 95% confidence intervals were reported.

Simulation II: Rote vs. Interacting Association

To better contrast the differences with traditional ANNs and their use of rote association, the current model was evaluated in parallel to the vanilla Multi-Layer Perceptron (MLP). Each model was presented the 2- to 9-bit in their respectful format and under fair conditions.

This meant that for the MLP, the task required learning and recalling a total of 2^N inputs. As usual, to accommodate the level increase, the number of units in the input layer of the MLP varied in function of N [Chen, Tseng & Wu, 2014]. Thus, for each level increase, the architecture had to be adjusted. The MLP was implemented in Scikit-learn [Pedregosa et al., 2011] with the parameters kept at their defaults values with the exception of the *batch_size*, which was set to 2^N to mimic being presented one stimulus at a time, the *solver*, which was set to "sgd" (stochastic gradient descent) to follow the error rate per epoch, and *learning_rate_init*, which was set to $1/(2^N)$ to minimize the number of learning epochs.

For the proposed model, the task format required learning only the interacting associations needed for a specific bit value and was evaluated during recall on that size. This restriction ensured a high-level comparison, as the MLP can only learn one level at a time. Thus, for each increase, the model was retrained on the selected associations. For example, for the 2-bit, the MF-BAMs were only taught the context associations, and the ones needed to count to 2 in Fig. 3. This meant that the model only had 12 associations in total to learn. However, during recall, it was presented all four times series representing the 2-bit parity task. Thus, for each bit size increase, three additional associations were needed during learning. In other words, for the 3-, 4-, 5-bit, and so on, 15, 18, 21, etc., associations were used, respectively.

To show the differences between approaches, learning was stopped after each MF-BAM or MLP reached a MSE < 0.005 for the N-bit task in question (indicating that the task was correctly learned). As such, the total number of epochs for each network was reported. To assess reliability, learning and recall were repeated 50 times for both models, and 95% confidence intervals were reported.

Simulation III: Noise Filtering

The model's capability to still filter noise like any other BAM was evaluated on the 2-bit time series through pixel flip noise, which was introduced to all patterns of the time series. This bit level was chosen for simplicity, as increasing N only lengthens the time series and does not change the nature of the task. An example of different degrees of noise can be seen in Fig. 5. The number of pixel flips varied from 0 to 35, giving a proportion of noise from 0 to 71 % (35/49). To ensure consistency, the model was trained on all input-target pairs shown in Fig. 3. For reliability purposes, 50 learning trials were conducted of 2500 epochs. However, 100 testing trials were performed for each learning run due to the random selection of the pixel to be flipped. As such, average performances and 95% confidence intervals were reported.

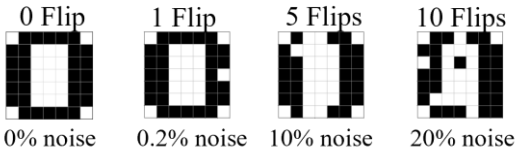

Fig. 5: Example of different levels of pixel flip.

Learning Parameters

For all simulations, each MF-BAM's learning was conducted in the same fashion as in [Rolon-Mérette, Rolon-Mérette & Chartier, 2023]. The transmission function parameter (δ) was set to 0.2, weights were randomly initialized ~U(-0.1,0.1), and the number of units (u) per layer was set to 100, which is about the size of the input patterns of the Extractor. To assess the complexity of the associations, the number of layers (l) ranged from 1 to 4 for simulation I and III and was kept constant at l = 4 for simulation II.

Recall Process

Once learning was completed, the network was then tested by presenting the time series given by the simulation condition (for an example of the N-bit as a time series, see Fig. 2). As previously mentioned, the model's final output (parity) is given once the times series ends with the P pattern. Performance was calculated in the same fashion for each task, where the generated parity was compared to the original target on a pixel-per-pix basis. If the generated and the target patterns were identical, then it was recorded as a success.

Section III: Results

Simulation I: The N-bit and Interacting Association

The model learned all interacting associations and was tested on the entirety of the times series representing the 2- to 9-bit. Fig. 6 shows the SL layer's MSE for both the Identifier and Extractor when learning its 4 and 29 input-target associations, respectively. Results indicated that as l increased, error decreased. This was more evident for the Extractor, where at the 500 epochs mark, l = 1 had an MSE of 0.02596, while for l = 4, it was 0.00079. For the Identifier, this was also observed. However, a major difference could only be seen for very small numbers of epochs (< 100). For example, at 25 epochs, l = 1 had an MSE of 0.025507 and 0.00617 for l = 4, while at 500 epochs, both were lower (MSE < 0.0001). Fig. 6 also shows that although learning continued for 2500 epochs, no spikes in error were observed for either network. Results also showed that for l = 1, the MSE values remained unchanged (≈ 0.0027) for the last 200 epochs, indicating that the model was stuck. All other layers converged at lower values (MSE < 0.0001).

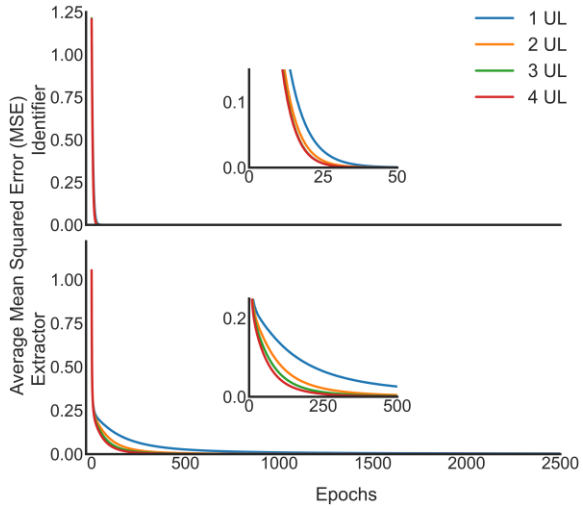

Fig 6: Average mean squared error (MSE) for the Identifier and Extractor under different *l* conditions.

Fig. 7 shows the average recall performance and 95% confidence intervals in function of the 2- to 9-bit parity task. Results showed that a single layer could only solve the 2-bit consistently. However, for deeper architectures ($l > 1$), all parity tasks could be learned systematically despite changes in the time series length.

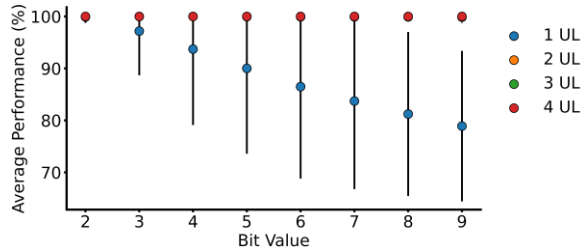

Fig. 7: Average recall performance and 95% confidence intervals for each bit size and *l* condition.

Simulation II: Rote vs. Interacting Association

The model was compared to a MLP, and learning was stopped when each network's MSE was < 0.005. Results showed that both had perfect recalls and that as the number of bits increased, the minimum learning time increased exponentially for the MLP (Fig. 8 a)). However, this was not the case for the MF-BAMs as the total learning times only slightly increased, going on average from around 700 epochs (2-bit) to almost 1400 (9-bit), thus showing a much more linear trend. 95% confidence intervals also showed that the later model fluctuated less. Fig. 8 b) showed that as the number of bits increased, the Extractor module learning time also increased. However, this was not the case for the Identifier, as no changes in learning times were observed. This was expected since the number of learned patterns did not change (i.e., the same four context associations).

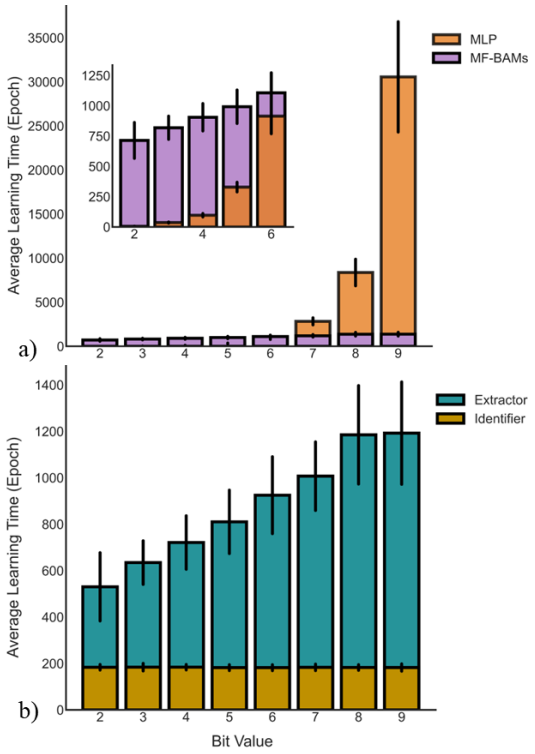

Fig. 8: Average learning times and 95% confidence intervals for each bit value a) MF-BAMs (Identifier + Extractor; purple) and MLP (orange). b) Identifier (sand) and Extractor (turquoise).

Simulation III: Noise Filtering

The effect of noisy inputs was studied using pixel flips for the 2-bit task only. Fig. 9 shows that as the number of flips increased, performances decreased. Interestingly, results show that even if a single layer ($l = 1$) was sufficient to solve it (Fig. 7), there is an increase in noise tolerance proportional to the number of layers.

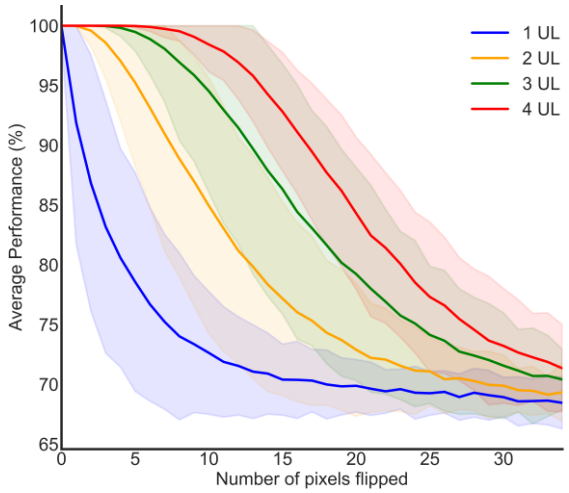

Fig. 9: Average pixel flip performances for all *l* conditions (full lines) and 95% confidence intervals (shaded area).

Section IV: Discussion

A recurrent neural associative memory composed of two modules showed that it was possible to solve any *N*-bit parity tasks (2-9) solely by associative mechanism. This was achievable by the model's capability to learn multiple interacting associations.

Because of some nonlinear nature between the 33 associations, it was expected that the architecture of each module had to contain multiple hidden layers to learn correctly. Results in Fig. 6 showed that to learn its 29 associations and ensure systematic performances (Fig. 7), an architecture of $l > 1$ was necessary for the Extractor. This was not the case for the Identifier. Learning times in Fig. 6 suggested that a shallower architecture would have sufficed, as there were no major differences between layer sizes. This can be attributed to the fact that the four associations were one-to-one, making them simpler and quicker to learn. Interestingly, this difference in learning speed between modules could be used to mimic a sort of short-term scenario where the Identifier would constantly be retrained with new directives. This would be particularly handy in situations where the nature of the tasks does not drastically change, for example, counting the number of 0s instead of 1s and determining the parity. In such an instance, just the directive of when to count or stay fixed needs to change. In other words, the Extractor does not need to be retrained. Thus, adapting to such a shift becomes only a question of a few epochs, as only the Identifier, the small and fast portion of the model, would have to be adjusted. Regardless, it is worth further investigating if different behaviors could be obtained by changing the architectural parameter of each module independently.

By using pattern representations for the *N*-bit task, the dimensionality of the inputs remained the same. This had three important consequences. First, no changes in architecture are required to accommodate a shift in level, as shown by Simulation I. Second, no bias units are needed anymore due to the nature of the task becoming associative. Third, the representation of the *N*-bit became much more realistic, as presenting time series is akin to having a human gaze upon a series of values one at a time. As such, the consequences of using pattern representations are of interest and would be worth maintaining for future tasks, especially from a cognitive perspective.

In addition, by using the model's recursive properties, varying *N* only impacted the number of inputs presented without affecting learning times. Results in simulation II showed that the learning times did not increase exponentially but rather linearly. Although requiring ≈1400 epochs is still quite lengthy for solving the 2- to 9-bit from a plausible perspective, past research has shown that increasing layer depth and width can lower learning times and increase performances [Rolon-Mérette, Rolon-Mérette & Chartier, 2023]. As such, it would be worth investigating which specific architectures results in the fastest and most accurate performances.

Of importance, when learning the correct associations and interactions, the objective no longer becomes to map the entire solution space but rather learn the necessary associations to solve the task. This is where this research distinguished itself, as it became more akin to how humans learn to perform these types of problems [Soto-Calvo et al., 2015] while remaining in a distributed bidirectional information signal processing approach. It showed that compared to rote learning, only 33 associations are needed rather than 1020. This becomes much more interesting from a cognitive perspective.

Results from simulation III showed that increasing the number of layers had a beneficial effect on noise tolerance. This is expected as MF-BAMs were not allowed to cycle their individual networks to clean the signal further. Therefore, the noise was filtered as the pattern went through each layer. Future results should look if there is a tradeoff between the number of cycles per layer and the number of layers. Accordingly, each module should be tweaked independently, as their roles seem complementary. Evidence in humans also points to this modular distinction [Barrett & Kurzban, 2006; Chandra et al., 2018; Fang et al., 2004].

Although it was shown that the *N*-bit could be solved regardless of its length, some improvements could be made. First, the contextual tags were explicitly given. A more interesting solution would be to have the model generate these tags by itself and then use them for learning [Rolon-Mérette, Rolon-Mérette & Chartier, 2018]. This would increase the plausibility of the approach. Furthermore, the information had to be carefully divided between the Identifier and Extractor during learning. An interesting solution would be that such division would be automatic and that the model could learn these online. Thus, a sort of self-specialization should occur without external intervention [Kohonen, 1990; Miljković, 2017].

It is worth highlighting that if solving a task greater than the 9-bit was desired, then by doubling the dimension of the patterns (2x49), the model could also solve up to 99-bit with the same approach and advantages. Of further importance, this implementation is not specific to *N*-bit. The same approach could be applied to more cognitive relative tasks, such as the Dimensional Card Change Sort [Zelazo, 2006] and the Wisconsin Card Sorting test [Miles et al., 2021].

In summary, by using associative mechanisms, it was possible to have different modules interact, and by simply teaching the appropriate associations, the desired behaviors could be obtained in a more generalized manner. As such, by following approaches that try to respect plausibility, new important steps toward better understanding human cognition through ANNs can be made.

Acknowledgement

This research was supported by the Fond de Recherche du Quebec Nature et Technologies (FRQNT) and the Ontario graduate scholarship (OGS) and was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) (210663).

References

Acevedo-Mosqueda, M. E., Yanez-Marquez, C., & Acevedo-Mosqueda, M. A. (2013). Bidirectional associative memories: Different approaches. *ACM Computing Surveys (CSUR)*, *45*(2), 1-30.

Anderson, J. A. (1983). Cognitive and psychological computation with neural models. *IEEE transactions on systems, man, and cybernetics*, (5), 799-815.

Anaconda Software Distribution. (2020). Anaconda Documentation. Anaconda Inc. Retrieved from https://docs.anaconda.com/

Anderson, J. A., Silverstein, J. W., Ritz, S. A., & Jones, R. S. (1977). Distinctive features, categorical perception, and probability learning: Some applications of a neural model. Psychological review, 84(5), 413.

Anderson, J. R., & Bower, G. H. (2014). Human associative memory. *Psychology press*.

Barrett, H. C., & Kurzban, R. (2006). Modularity in cognition: framing the debate. *Psychological review*, *113*(3), 628.

Berch, D. B., Foley, E. J., Hill, R. J., & Ryan, P. M. (1999). Extracting parity and magnitude from Arabic numerals: Developmental changes in number processing and mental representation. *Journal of experimental child psychology*, *74*(4), 286-308.

Buckner, R. L., & DiNicola, L. M. (2019). The brain's default network: updated anatomy, physiology and evolving insights. *Nature Reviews Neuroscience*, *20*(10), 593-608.

Chandra, R., Gupta, A., Ong, Y. S., & Goh, C. K. (2018). Evolutionary multi-task learning for modular knowledge representation in neural networks. *Neural Processing Letters*, *47*, 993-1009.

Chartier, S., & Boukadoum, M. (2006). A bidirectional heteroassociative memory for binary and grey-level patterns. *IEEE Transactions on Neural Networks*, *17*(2), 385-396.

Chartier, S., Giguère, G., Renaud, P., Lina, J. M., & Proulx, R. (2007, August). FEBAM: A feature-extracting bidirectional associative memory. In *2007 International Joint Conference on Neural Networks* (pp. 1679-1684). IEEE.

Church, K., Ross, M., & Chartier, S. (2020). Using a Bidirectional Associative Memory and Feature Extraction to model Nonlinear Exploitation Problems. *In Proceedings of the 18th International Conference on Cognitive Modelling* (pp. 37-43).

Cichy, R. M., & Kaiser, D. (2019). Deep neural networks as scientific models. *Trends in cognitive sciences*, *23*(4), 305-317.

Darcey, M. (2016) Rethinking associations in psychology. *Synthese*, *193*(12), 3763-3786

Dacey, M. (2020), Associationism in the Philosophy of Mind. The Internet Encyclopedia of Philosophy, https://iep.utm.edu/associat/#H7.

Dhar, V. K., Tickoo, A. K., Koul, R., & Dubey, B. P. (2010). Comparative performance of some popular artificial neural network algorithms on benchmark and function approximation problems. *Pramana*, *74*, 307-324.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179-211.

Fang, C. Y., Fuh, C. S., Yen, P. S., Cherng, S., & Chen, S. W. (2004). An automatic road sign recognition system based on a computational model of human recognition processing. *Computer vision and Image understanding*, *96*(2), 237-268.

Graham, T. A. (1999). The role of gesture in children's learning to count. Journal of Experimental Child Psychology, 74(4), 333-355.

Jordan, M. I. (1997). Serial order: A parallel distributed processing approach. *In Advances in psychology* (Vol. 121, pp. 471-495). North-Holland.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, *78*(9), 1464-1480.

Kosko, B. (1988). Bidirectional associative memories. *IEEE Transactions on Systems, man, and Cybernetics*, *18*(1), 49-60.

Knoblauch, A. (2005). Neural associative memory for brain modeling and information retrieval. *Information Processing Letters*, *95*(6), 537-544.

Lansner, A. (2009). Associative memory models: from the cell-assembly theory to biophysically detailed cortex simulations. *Trends in neurosciences*, *32*(3), 178-186.

Mandelbaum, E. (2015b). Associationist theories of thought. In Zalta, E. N. (Ed.) *The Stanford encyclopedia of philosophy*.

Marcus, G. (2018). Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

Martindale, C. (1991). Cognitive psychology: A neural-network approach. Thomson Brooks/Cole Publishing Co.

Marvin, M., & Seymour, A. P. (1969). Perceptrons. Cambridge, MA: MIT Press, 6, 318-362.

Miles, S., Howlett, C. A., Berryman, C., Nedeljkovic, M., Moseley, G. L., & Phillipou, A. (2021). Considerations for using the Wisconsin Card Sorting Test to assess cognitive flexibility. Behavior research methods, 53(5), 2083-2091.

Miljković, D. (2017, May). Brief review of self-organizing maps. *In 2017 40th international convention on information and communication technology, electronics and microelectronics (MIPRO)* (pp. 1061-1066). IEEE.

O'Reilly, R. C. (1998). Six principles for biologically based computational models of cortical cognition. *Trends in cognitive sciences*, *2*(11), 455-462.

Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830.

Rolon-Mérette, D., Rolon-Mérette, T., & Chartier, S. (2019). Learning and Recalling Arbitrary Lists of Overlapping Exemplars in a Recurrent Artificial Neural Network. *In Proceedings of the 17th International Conference on Cognitive Modelling* (pp. 186-191).

Rolon-Mérette, D., Rolon-Mérette, T., & Chartier, S. (2023). A Multi-Layered Associative Memory Model for Learning Non-Linear Tasks. Available at SSRN 4259879.

Rolon-Merette, T., Rolon-Merette, D., & Chartier, S. (2018). Generating Cognitive Context with Feature-Extracting Bidirectional Associative Memory. *Procedia computer science*, *145*, 428-436.

Rolon-Mérette, D., Ross, M., Rolon-Mérette, T., & Church, K.

(2020). Introduction to Anaconda and Python: Installation and setup. Quant. Methods Psychol, 16(5), S3-S11.

Soto-Calvo, E., Simmons, F. R., Willis, C., & Adams, A. M. (2015). Identifying the cognitive predictors of early counting and calculation skills: Evidence from a longitudinal study. *Journal of Experimental Child Psychology*, *140*, 16-37.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *nature*, *529*(7587), 484-489.

Tesauro, G., & Janssens, B. (1988). Scaling relationships in back-propagation learning. Complex Systems, 2(1), 39-44.

Wang, J. H., & Cui, S. (2018). Associative memory cells and their working principle in the brain. *F1000Research*, 7.

Yanling, Z., Bimin, D., & Zhanrong, W. (2002, November). Analysis and study of perceptron to solve XOR problem. *In The 2nd International Workshop on Autonomous Decentralized System, 2002*. (pp. 168-173). IEEE.

Yang, J., Yang, W., & Wu, W. (2011). A novel spiking perceptron that can solve XOR problem. *Neural Network World*, *21*(1), 45.

Zador, A. M. (2019). A critique of pure learning and what artificial neural networks can learn from animal brains. *Nat Commun 10* (1): 1–7.

Zelazo, P. D. (2006). The Dimensional Change Card Sort (DCCS): A method of assessing executive function in children. *Nature protocols*, *1*(1), 297-301.