Abstract—Using the reservoir computing framework, we demonstrate how a simple model can solve an alternation task without an explicit working memory. To do so, a simple bot equipped with sensors navigates inside a 8-shaped maze and turns alternatively right and left at the same intersection in the maze. The analysis of the model's internal activity reveals that the memory is actually encoded inside the dynamics of the network. However, such dynamic working memory is not accessible such as to bias the behavior into one of the two attractors (left and right). To do so, external cues are fed to the bot such that it can follow arbitrary sequences, instructed by the cue. This model highlights the idea that procedural learning and its internal representation can be dissociated. If the former allows to produce behavior, it is not sufficient to allow for an explicit and fine-grained manipulation.

Index Terms—reservoir computing, robotics, simulation, working memory, procedural learning, implicit representation, explicit representation.

I. INTRODUCTION

Suppose you want to study how an animal, when presented with two options A and B, can learn to alternatively choose A then B then A, etc. One typical lab setup to study such alternate decision task is the T-maze environment where the animal is confronted to a left or right turn and can be subsequently trained to display an alternate choice behavior. This task can be easily formalized using a block world as it is regularly done in the computational literature. Using such formalization, a simple solution is to negate (logically) a one bit memory each time the model reaches A or B such that, when located at the choice point, the model has only to read the value of this memory in order to decide to go to A or B. However, as simple as it is, this abstract formalization entails the elaboration of an explicit internal representation keeping track of the recent behavior, implemented in a working memory that can be updated when needed. But then, what could be the alternative? Let us consider a slightly different setup where the T-Maze is transformed into a closed 8-Maze (see figure 1-Left). Suppose that you can only observe the white area when the animal is evolving along the arrowed line (both in observable and non-observable areas). From the observer point of view, the animal is turning left one time out of two and turning right one time out of two. Said differently, the observer can infer an alternating behavior because of its partial view of the system. The question is: does the animal really implement an explicit alternate behavior or is it merely following a mildly complex dynamic path? This is not a rhetorical question because depending on your hypothesis, you may search for neural correlates that actually do not exist. Furthermore, if the animal is following such mildly complex dynamic path, does this mean that it has no explicit access to (not to say no consciousness of) its own alternating behavior?

This question is tightly linked to the distinction between implicit learning (generally presented as sub-symbolic, associative and statistics-based) and explicit learning (symbolic, declarative and rule-based). Implicit learning refers to the non-conscious effects that prior information processing may exert on subsequent behavior [1]. It is implemented in associative sensorimotor procedural learning and also in model-free reinforcement learning, with biological counterparts in the motor and premotor cortex and in the basal ganglia. Explicit learning is associated with consciousness or awareness, and to the idea of building explicit mental representations [2] that can be used for flexible behavior, involving the prefrontal cortex and the hippocampus. This is what is proposed in model-based reinforcement learning and in other symbolic approaches for planning and reasoning. These strategies of learning are not independent but their relations and interdependencies are not clear today. Explicit learning is often observed in the early stages of learning whereas implicit learning appears on the long run, which can be explained as a way to decrease the cognitive load. But there is also a body of evidence,
for example in sequence learning [3] or artificial grammar learning studies [4], that suggests that explicit learning is not a mandatory early step and that improvement in task performance are not necessarily accompanied by the ability to express the acquired knowledge in an explicit way [5].

Coming back to the task mentioned above, it is consequently not clear if we can learn rules without awareness and then to what extent can such implicit learning be projected to performance in an unconscious way? Furthermore, without turning these implicit rules into an explicit mental representation, is it possible to manipulate the rules, which is a fundamental trademark of flexible adaptable control of behavior?

Using the reservoir computing framework generally considered as a way to implement implicit learning, we first propose that a simple alternation or sequence learning task can be solved without an explicit pre-encoded representation of memory. However, to then be able to generate a new sequence or manipulate the rule learnt, we explain that inserting explicit cues in the decision process is needed. In a second series of experiments, we provide a proof of concept still using the reservoir computing framework, for the hypothesis that the recurrent network forms contextual representations from implicitly acquired rules over time. We then show that these representations can be considered explicit and necessary to be able to manipulate behaviour in a flexible manner.

In order to provide preliminary interpretation of what is observed here, it is reminded that recurrent networks, particularly models using the reservoir computing framework, are a suitable candidate to model the prefrontal cortex [6], also characterized by local and recurrent connections. Given their inherent sensitivity to temporal structure, it also makes these networks adaptable for sequence learning. This approach has been used to model complex sensorimotor couplings [7] from the egocentric view of an agent (or animal) that is situated in its environment and can autonomously demonstrate reactive behaviour from its sensory space [8], as we also do in the first series of experiments, for learning sensorimotor couplings by demonstration, or imitation. In the second series of experiment, we propose that the prefrontal cortex is the place where explicit representations can be elaborated when flexible behaviors are required.

II. METHODS AND TASK

The objective is the creation of a reservoir computing network of type Echo State Network (ESN) that controls the movement of a robot [8], [9], which has to solve a decision-making task (alternately going right and left at an intersection) in the maze presented in figure [1].

A. Model Architecture: Echo State Network

An ESN is a recurrent neural network (called reservoir) with randomly connected units, associated with an input layer and an output layer, in which only the output (also called readout) neurons are trained. The neurons have the following dynamics:

\[
x[n] = (1 - \alpha)x[n - 1] + \alpha \tilde{x}[n] \quad (1)
\]

\[
\tilde{x}[n] = \tanh(W_x[n - 1] + W_{in}[1; u[n]]) \quad (2)
\]

\[
y[n] = f(W_{out}[1; \tilde{x}(n)]) \quad (3)
\]

where \(x(n)\) is a vector of neurons activation, \(\tilde{x}(n)\) its update, \(u(n)\) and \(y(n)\) are respectively the input and the output vectors, all at time \(n\). \(W, W_{in}, W_{out}\) are respectively the reservoir, the input and the output weight matrices. The notation \([; ;]\) stands for the concatenation of two vectors. \(\alpha\) corresponds to the leak rate, \(\tanh\) corresponds to the hyperbolic tangent function and \(f\) to linear or piece-wise linear function.

The values in \(W, W_{in}, W_{out}\) are initially randomly chosen. While \(W, W_{in}\) are kept fixed, the output weights \(W_{out}\) are the only ones plastic (red arrows in Figure [2]). In this model, the output weights are learnt with the ridge regression method (also known as Tikhonov regularization):

\[
W_{out} = Y_{\text{target}}X^T (XX^T + \beta I)^{-1} \quad (4)
\]

where \(Y_{\text{target}}\) is the target signal to approximate, \(X\) is the concatenation of \(1\), the input and the neurons activation vectors: \([1; u(n); x(n)]\), \(\beta\) corresponds to the regularization coefficient and \(I\) the identity matrix.

B. Experiment 1: Uncued sequence learning

The class of tasks called spatial alternation has been widely used to study hippocampal and working memory functions [10]. For the purpose of our investigation, we simulated a continuous version of the same task, wherein the agent needs to alternate its choice at a decision point, and after the decision, it is led back to the central corridor, in essence following an 8-shaped trace while moving (see figure [1] Left). This alternation task is widely believed to require a working memory such as to remember what was the previous choice in order to alternate it. Here we show that the ESN previously described is sufficient to learn the task without an explicit representation of the memory.

1) Tutor model: In order to generate data for learning, we implemented a simple Braininenberg vehicle where the robot moves automatically with a constant speed and changes its orientation according to the values of its sensors. At each time step the sensors measure the distance to the walls and the bot turns such as to avoid the walls. At each timestep, the position of the bot is updated as follows:

\[
\theta(n) = \theta(n - 1) + 0.01 \sum_i \alpha_i s_i \quad (5)
\]

\[
p(n) = p(n - 1) + 2 \times (\cos(\theta(n)) + \sin(\theta(n))) \quad (6)
\]

where \(p(n)\) and \(p(n + 1)\) are the positions of the robot at time step \(n\) and \(n + 1\), \(\theta(n)\) is the orientation of the robot, calculated as the weighted sum (\(\alpha_i\)) of the values of the sensors \(s_i\). The norm of the movement is kept constant and fixed at 2.
The ESN was built with the python library ReservoirPy [11] with the hyper-parameters presented in table I column “Without context”. The order of magnitude of the hyper-parameters was first found using the Hyperopt python library [12], then these were fine tuned manually. The ESN receives as input the values of the 8 sensors and output the next orientation.

4) Model evaluation: The performance of the ESN has been calculated with the Normalized Root Mean Squared Error metrics ($NRMSE$) and the R square ($R^2$) metrics, defined as follows:

$$NRMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{n}}$$

(7)

$$R^2 = 1 - \frac{\sum(y_i - \bar{y})^2}{\sum(y_i - \overline{\hat{y}})^2}$$

(8)

where $y_i$, $\hat{y}_i$ and $\overline{\hat{y}}$ are respectively the desired output, the predicted output and the mean of the desired output.

5) Reservoir state analysis: In this section the reservoir states are analysed such as to inspect to which extent they form an internal and hidden representation of the memory. To do so, we use Principal Component Analysis (PCA), a dimensionality reduction method enabling the identification of patterns and important features of the processed data. PCA is carried out on the reservoir states for each position of the robot during the 8-shape trajectory. We continued the analysis by doing a classification of the reservoir states. We made the assumption that it is possible to know the future direction of the robot observing the internal states of the reservoir. This implies that the reservoir states can be classified in two classes: one related to the prediction of going left and the other related to the prediction of going right. Two standard classifiers, the KNN (K-Nearest Neighbors) and the SVM (Support Vector Machine) were used. They take as input the reservoir state at each position of the bot while executing the 8-shape and predict the decision the robot will take at the next intersection. Since the Braitenberg algorithm only aims to avoid obstacles, if the robot is not in the left or right pathway, the added walls are removed and the true position values of the robot at every time step in order to make the training more robust. Approximately 50,000 time steps were generated (equivalent to 71 complete 8-loops) and separated into training and testing sets.

2) Training data: The ESN is trained using supervised learning, containing samples from the desired 8-shaped trajectory. Since the Braitenberg algorithm only aims to avoid obstacles, the robot is forced into the desired trajectory by adding walls at the intersection points as shown in figure 3. After generating the right pathway, the added walls are removed and the true sensor values are gathered as input. Gaussian noise is added to the position values of the robot at every time step in order to make the training more robust. Approximately 50,000 time steps were generated (equivalent to 71 complete 8-loops) and separated into training and testing sets.

3) Hyper parameters tuning: The ESN was built with the python library ReservoirPy [11] with the hyper-parameters presented in table I column “Without context”. The order
C. Experiment 2: 8 Maze Task with Contextual Inputs

In this experiment, we fed the reservoir with two additional inputs that represent the next decision, one being related to a right turn (R) and the other to a left turn (L) (see figure 2). They are binary values, switched to a value of 1 only when the bot is known to take the corresponding direction. We thus built a second ESN with the hyper-parameters presented in TABLE I, column “With context”. The network is similar to the previous one, except that the contextual inputs are added with a different input scaling than the one used for the sensors inputs. During the data generation, the two additional inputs are set to 0 everywhere in the maze, except in the central corridor.

III. RESULTS

A. Motor sequence learning

We first show that a recurrent neural network like the ESN can learn a rule-based trajectory in the continuous space, without an explicit memory or feedback connections. The score of the ESN is shown in TABLE II and the results for the trajectory predicted by the ESN are presented in figure 4 and in the top panel in figure 8. At each period of about 350 steps, a behavior or decision switch takes place, which is evident from the crests and troughs in the y-axis coordinates. It can be seen that the ESN correctly predicts the repeated alternating choice in the central arm of the maze. In addition to switching between the left and right loops, the robot also moves through the environment without colliding into obstacles.

| Performance of the ESN for 50 simulations |
|-------------------------------------------|
| NRMSE | $R^2$ |
|-------|-------|
| Mean  | 0.0171| 0.9962|
| Variance | 5.4466e-06 | 1.0192e-06 |

TABLE II

NRMSE and $R^2$ SCORE OF THE ESN WITH 8 INPUTS

B. Reservoir State Prediction

Next, we show that even a simple classifier such as SVM or KNN can observe the internal states of the reservoir and learn to predict the decision (whether to go left or right) of the network. The results of the predictions are presented in the top part of figure 5. As expected, there is a periodicity of choice in line with the position of the bot in the maze, showing that the classification is relevant. At each time step, both classifiers output the same prediction with a small discrepancy in time. The accuracy score obtained for both classifiers is 1. In the bottom part of figure 5, we can observe that the robot knows quite early which decision it will take at the next loop while we could expect that it would take its decision in the yellow corridor in figure 1. Here, we see that if the robot just turned right, the reservoir switches its internal state to go left next time only a few dozen time steps after. We tried the same classifiers but instead of the reservoir states as input, we used the sensors values. The results are shown in the figure 5. As expected, the classifications fail with an accuracy score of 0.57 for SVM and 0.43 for KNN; this randomness can be seen in both figures. Thus, we showed that by simply observing the internal states of the reservoir, it is possible to predict its next prediction. In essence, this is a proof of concept to show that second-order or observer networks, mimicking the role of the regions of the prefrontal cortex implementing contextual rules, can consolidate information linking sensory information to motor actions, to develop relevant contextual representations.

Since the state space of the dynamic reservoir is high-dimensional, using the Principal Component Analysis (PCA) on the states, we investigated if it is possible to observe sub-space attractors. The result for the PCA analysis is presented in figure 7, where PCA was applied for 5000 time steps, which corresponds to 7 8-loops. The figure shows two symmetric sub-attractors, which are linearly separable, that correspond to the two parts of the 8-shape trajectory.

C. Explicit rules with contextual inputs

Finally, we demonstrate that although the ESN can learn a motor sequence without contextual inputs, it is limited by its internal representation to learn more complex sequences which may require a longer memory. Adding contextual or explicit information about the rule (which we propose are representations developed by the prefrontal cortex over time) can then bias the ESN to follow any arbitrary trajectory as in 8. With the additional contextual inputs, the ESN is able to reproduce the standard 8 sequence (the performance is shown in table III) but can also achieve more complex tasks by sending to it the proper contextual inputs. One example can be seen in figure 8: the top graph shows the positions of the bot while making the standard 8 sequence [ABABABAB...], the bottom one shows that the bot was able to achieve a more complex sequence [AABBABBAABB...].

IV. DISCUSSION

Using a simple reservoir model that learns to follow a specific path, we have shown how the resulting behavior could be
interpreted as an alternating behavior by an external observer. However, we’ve also shown that from the point of view of the model and in the absence of associated cues, this behavior cannot be interpreted as such. Instead, the behavior results from the internal dynamics of the reservoir (and the learning procedure we implemented). Without external cues, the model is unable to escape its own behavior and is trapped inside an attractor. Only the cues can provide the model with the necessary and explicit information that in turn allows to bias its behavior in favor of option A or option B.

From a neuroscience perspective, as developed in more details in [13], it can be proposed that the reservoir model in the first experiment implements the premotor cortex learning sensorimotor associations in the anterior cortex. In the first experiment, this is made by supervised learning in a process of learning by imitation. In a different protocol, this is also classically be done by reinforcement learning, involving another region of the anterior cortex, the anterior cingulate cortex.
manipulating prediction of the outcome. Whereas both regions of the anterior cortex and present in mammals, [13] reports that another region, the lateral prefrontal cortex, is unique in primates and has been developed to implement the learning of contextual rules and to possibly act in a hierarchical way in the control of the other regions. We have proposed an elementary implementation of the lateral prefrontal cortex in the second experiment, adding explicit contextual inputs as a basis to form contextual rules. It was accordingly very important to observe that it was then possible to explicitly manipulate the rules and form flexible behavior, whereas in the previous case, rules were implicitly present in the memory but not manipulable.

This simple model shows that the interpretation of the behavior by an observer and the actual behavior might greatly differ even when we can make accurate prediction about the behavior. Such prediction can be incidentally true without actually revealing the true nature of the underlying mechanisms. Based on the reservoir computing framework which can be invoked for both premotor and prefrontal regions, we have implemented models which are structurally similar (as it is the case for that regions) and we have shown that a simple difference related to their inputs can orient then toward implicit or explicit learning as respectively observed in the premotor and lateral prefrontal regions. It will be important in future work to see how these regions are associated to combine both modes of learning and switch from on to the other depending on the complexity of the task.

REFERENCES

[1] A. Cleeremans, “Implicit learning and implicit memory,” Encyclopedia of Consciousness, p. 369–381, 2009. [Online]. Available: http://dx.doi.org/10.1016/B978-012273873-3.00047-5

[2] A. Cleeremans, B. Timmermans, and A. Pasquali, “Consciousness and metarepresentation: A computational sketch,” Neural Networks, vol. 20, no. 9, pp. 1032–1039, 2007.

[3] B. A. Clegg, G. J. DiGirolamo, and S. W. Keele, “Sequence learning,” Trends in cognitive sciences, vol. 2, no. 8, pp. 275–281, 1998.

[4] A. S. Reber, Implicit Learning and Tacit Knowledge. Oxford University Press, Sep 1996. [Online]. Available: http://dx.doi.org/10.1093/acprof:oso/9780195106589.001.0001

[5] Z. Dienes and D. Berry, “Implicit learning: Below the subjective threshold,” Psychonomic bulletin & review, vol. 4, no. 1, pp. 3–23, 1997.

[6] P. Dominey, M. Arbib, and J.-P. Joseph, “A model of corticostriatral plasticity for learning oculomotor associations and sequences,” Journal of cognitive neuroscience, vol. 7, no. 3, pp. 311–336, 1995.

[7] J. Tani and S. Nolfi, “Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems,” Neural Networks, vol. 12, no. 7-8, pp. 1131–1141, 1999.

[8] E. Aislan Antonelo and B. Schrauwen, “On learning navigation behaviors for small mobile robots with reservoir computing architectures,” IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 4, p. 763–780, Apr 2015. [Online]. Available: http://dx.doi.org/10.1109/TNNLS.2014.2323247

[9] E. Antonelo and B. Schrauwen, “Learning slow features with reservoir computing for biologically-inspired robot localization,” Neural Networks, vol. 25, p. 178–190, Jan 2012. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2011.08.004

[10] L. M. Frank, E. N. Brown, and M. Wilson, “Trajectory encoding in the hippocampus and entorhinal cortex,” Neuron, vol. 27, no. 1, pp. 169–178, 2000.

[11] N. Trouvain, L. Pedrelli, T. T. Dinh, and X. Hinaut, “ReservoirPy: An efficient and user-friendly library to design echo state networks,” in Artificial Neural Networks and Machine Learning – ICANN 2020. Springer International Publishing, 2020, pp. 494–505. [Online]. Available: https://doi.org/10.1007/978-3-030-61681-8_40

[12] J. Bergstra, B. Kommer, C. Elia samith, D. Yamins, and D. D. Cox, “Hyperopt: a python library for model selection and hyperparameter optimization,” Computational Science & Discovery, vol. 8, no. 1, p. 014008, 2015.

[13] E. Koechlin, “An evolutionary computational theory of prefrontal executive function in decision-making,” Philosophical transactions of the Royal Society of London. Series B, Biological sciences, vol. 369, no. 1655, pp. 20 130 474+, Nov. 2014. [Online]. Available: http://dx.doi.org/10.1098/rstb.2013.0474