A Appendix

A.1 Place cells

All agents have 49 place cells whose activity is a Gaussian function of the agent's position in the maze according to

$$u^{pc} = e^{-\frac{(x(t)-x_i)^2}{2\sigma_{pc}^2}} \tag{13}$$

where $x(t)$ indicates the agents position in the square maze bounded by $x = (0.8m, 0.8m)$. $\sigma_{pc} = 0.267m$ and place cells are spaced regularly apart in a 7 by 7 grid, while covering the boundary. The sensory cue passed to the agents is encoded by u^{cue}, which is a one-hot encoded vector of length 18 with gain 3. The cue vector is presented to the agent throughout the trial period. The place cell activity and the sensory cue is concatenated to form the input vector $Q(t)$ to all agents

$$Q(t) = [u^{pc}, u^{cue}] \tag{14}$$

A.2 Actor

All agents have $M = 40$ actor units where each unit represents a spatial direction. The firing rate of each actor unit is $\rho(t) = \text{ReLU}[q(t)]$ and the membrane potential q has dynamics

$$q(t) = (1-\alpha)q(t-1) + \alpha\left(\phi(t-1) + \sum W^{lateral}\rho(t-1) + \frac{\sigma_{actor}}{\sqrt{\alpha}}N(0,1)\right) \tag{15}$$

with $\alpha = \frac{100}{150} = 0.667$ and $\sigma_{actor} = 0.25$. $\phi(t)$ represents the output from the motor controller module. The lateral synaptic weight is given by

$$W^{lateral} = \frac{w_-}{M} + w_+\frac{f(k,h)}{\sum f(k,h)} \tag{16}$$

with $f(k,h) = (1-\delta)e^{\psi cos(\theta-\theta)}$, $w_- = -1$, $w_+ = 1$ and $\psi = 20$. The lateral connectivity connects the actor units into a ring attractor that smoothens the agent's spatial trajectory as in [26, 27]. The direction of movement is chosen by

$$a(t) = \frac{1}{M}\sum \rho(t)K^{dir} \tag{17}$$

which is the vector sum of directions weighed by each actor unit's firing rate with $K^{dir} = a_0(\sin\theta, \cos\theta)$ and $a_0 = 0.03$.

A.3 Advantage Actor-Critic

The Advantage Actor Critic (A2C) agent chooses one direction of movement $\rho^{a2c}(t)$ out of K^{dir} based on a stochastic action policy. To match the same speed achieved by the other agents, $a_0 = 0.07$ was chosen to increase the step size of a particular action in K^{dir}. The direction of movement is smoothed using a low pass filter

$$a(t) = (1-\alpha_{a2c})\widehat{a}(t) + \alpha_{a2c}\rho^{a2c}(t) \tag{18}$$

with $\alpha_{a2c} = 0.25$ as how Foster, Dayan & Morris (2000) [14] smoothened the trajectory of the actor-critic agent that chose discrete actions. The Advantage Actor-Critic reinforcement algorithm was implemented as in [1, 18, 19] where the agent is allowed to run through an entire trial, storing the various state, reward and actions taken before the weights are updated. Instead of the asynchronous method, only one CPU thread was used to run the synchronous method for each agent. The gradient

is computed at the end of each trial according to a weighted sum of the policy π, value function V and entropy regularisation term H according to

$$\begin{aligned}\nabla L &= \nabla L_\pi + \nabla L_v + \nabla L_{ent} \\ &= \frac{\partial \log \pi(a(t)|s(t),\theta)}{\partial \theta}\delta(t) + \\ \beta_v \delta(t)\frac{\partial V}{\partial \theta_v} &+ \beta_e \left(\frac{\partial H(\pi(a(t)|s(t),\theta))}{\partial \theta}\right) \\ \delta(t) &= [R^{disc}(t) - V(s(t),\theta_v)] \\ R^{disc}(t) &= \sum_{t=0}^{T} \gamma^{t-1} r(t) \end{aligned} \tag{19}$$

where θ and θ_v are the synaptic weight parameters for the policy and value function, and $\gamma = 0.99$ is the reward discount factor. Hyperparameters $\beta_v = 0.5$ and $beta_e = -0.001$ control the value estimate loss and entropy regularisation term contributions to the total loss. Lastly, $\delta(t)$ is the temporal difference error that informs the actor-critic of the advantage incurred. The critic is a single linear unit, actor is softmax activated and the hidden layer has 8192 units with a ReLU variant activation function where if the unit activity is above a threshold value of 3, the value is retained otherwise, it is converted to a 0. Using this variant showed a faster convergence in training compared to original ReLU function. Synaptic weights are updated using the RMSprop optimiser with learning rate 0.000035.

A.4 Reward disbursement

Each agent is free to explore the arena till the trial ends but if it finds the reward before, the agent remains stationary at the reward location until the trial ends to model consummatory behaviour. After the agent reaches the reward, a total reward value $R = 4$ is disbursed at a reward rate $r(t)$, similar to [26], given by

$$\begin{aligned} r_a(t) = (1 - \frac{100}{\tau_a})r_a(t-1), \quad r_b(t) = (1 - \frac{100}{\tau_b})r_b(t-1), \\ r(t) = \frac{r_a(t) - r_b(t)}{\tau_a - \tau_b} \end{aligned} \tag{20}$$

with $\tau_a = 120ms$ and $\tau_b = 250ms$ for all agents except for the Reservoir agent trained by sparse learning signal where $\tau_b = 2500ms$. When the agent reaches the reward, it is updated according to

$$r_a(t) \to r_a(t) + R \qquad r_b(t) \to r_b(t) + R \tag{21}$$

such that $r(t)$ integrates to R. In trials where the agent does not reach the reward location, no punishment is given, except for the A2C agent where a negative reward $R = -1$ is given to penalise the actions taken, else the agent converges to a stationary action policy.

A.5 Motor Controller

The symbolic motor controller is formulated by taking the goal $g(t)$ and current position $p(t)$ coordinates as inputs and performing vector subtraction.

$$Q^{vecsub}(t) = g(t) - p(t) \tag{22}$$

The dot product between the resultant vector $Q^{vecsub}(t)$ and K^{dir}

$$A^{mc}(t) = \text{softmax}(\beta_{mc} Q^{vecsub} K^{dir}) \tag{23}$$

with $\beta_{mc} = 4$ as a scaling factor, specifies the direction and magnitude in which the agent needs to move in order to reach the goal location. Using a softmax for $A^{mc}(t)$ creates a firing rate profile similar to the ring attractor where actions besides the optimum can also be taken to reach the goal location.

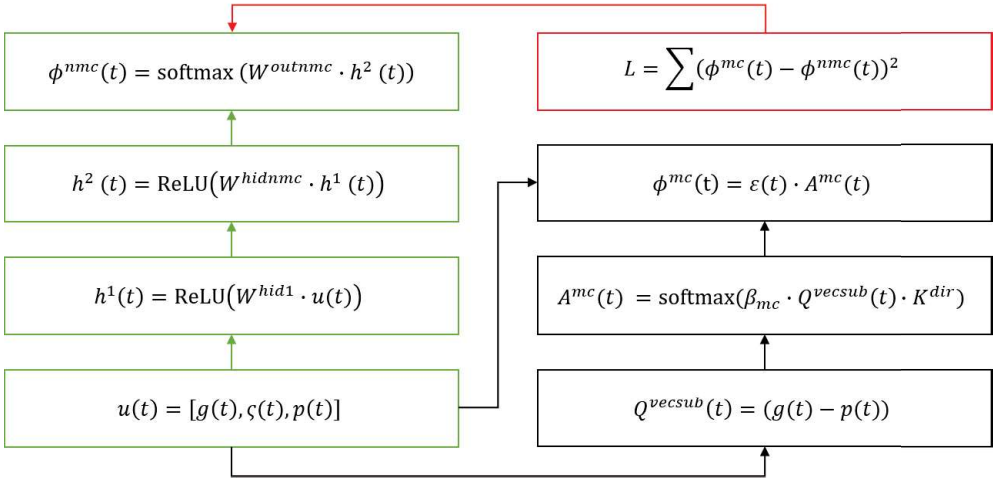

Figure 4: **Symbolic and neural motor controller architecture.** Goal and current position coordinates are taken in as inputs. In the symbolic motor controller, vector subtraction is performed and the direction of movement is chosen. The output of the symbolic motor controller is gated using $\varepsilon(t)$. A neural motor controller can be trained by backpropagation by taking the mean squared error between the symbolic motor controller target output ϕ^{mc} and neural motor controller output ϕ^{nmc}. The output of the neural motor controller does not need to be explicitly suppressed.

To allow the agent to freely explore the maze and find the target location versus turning on the motor controller for directed movement, a gating factor $\varepsilon(t)$ is used to signal the presence or absence of a goal and modulate the output of the symbolic controller ϕ^{mc} before passing it to the ring attractor in Eq. 15.

$$\phi^{mc}(t) = \varepsilon(t) A^{mc}(t) \tag{24}$$

We have formulated two solutions to gate the motor controller activity. The first involves the L_2 norm of the goal coordinate

$$\varepsilon(t) = \begin{cases} 1, & \text{if } \|g(t)\|^2 > \omega \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

If the L_2 norm is lesser or greater than the threshold ω, the output of the motor controller $A^{mc}(t)$ will or will not be suppressed by ε respectively. Such a control mechanism is similar to [14] where the agent freely explores the maze if the agent's memory was empty and performs goal directed movement if there was a stored goal coordinate. Instead, the episodic memory bank matrix used by the symbolic agent is initialised to 0 and the goal coordinate output by the reservoir is 0 since the synaptic weights W^{out} is initialised as 0.

The main results were obtained by using $\omega = 0.15$. This increased the propensity for the agents to explore the periphery of the maze but it also meant that the goal location with coordinates (0,0) could not be considered as a target. Instead, a lower threshold $\omega = 0.025$ can be used such that the coordinate (0,0) can be considered as goal. This is because the agent only stores the coordinate at which it hits the target, which is ± 0.03 from the center of the goal coordinate instead of the center of the goal coordinate. Lowering the threshold increases the agent's propensity to employ the motor controller and move to the center of the maze, especially when it encounters a new input $Q(t)$. For

example, an unforeseen input e.g. Cue 7 to Cue 18 during single session NPA training causes both the attention mechanism in the episodic memory bank and the reservoir to recall a goal coordinate vector with L_2 norm closer to (0,0). This causes the agent to move to the center first and searching for goal locations. This could be a similar strategy animals or humans employ. However, this limits the agent's exploration mostly to the center. The total trial time needs to be increased to 3600 seconds to cater more time for the agent to eventually find goal locations near the periphery of the maze.

The second solution involves learning a confidence function ς. The intuition is similar to the confidence ascribed to a recalled goal coordinate where a value closer to 1 indicates the highest confidence.

$$\varepsilon(t) = \begin{cases} 1, & \text{if } \varsigma > \omega \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

If ς is lesser or greater than the threshold ω, the output of the motor controller $A^{mc}(t)$ will or will not be suppressed by ε respectively.

Learning this confidence signal in the Symbolic agent involves storing $\varsigma = 1$ together with the current position coordinates $(p(t), \varsigma = 1)$ in a 18 X 3 Value matrix of the episodic memory bank, when the agent reaches the reward location. During recall, $(g(t), \varsigma)$ is obtained using the attention mechanism and passed to the symbolic motor controller together with the current position coordinates $p(t)$ of the agent.

Learning this confidence function ς by the Reservoir agents is similar to learning the target coordinates. Instead of two readout units, Reservoir agents now have three readout units, each to learn and represent $(g(t), \varsigma)$. The target to learn then becomes $(p(t), \varsigma = 1)$ for both the reservoir trained by the Perceptron (Eq. 8 and the exploratory Hebbian (Eq. 10 and Eq. 12) rules.

Similar algorithms can be either handcrafted or learnt by the agent to switch between these explore-exploit modalities. Unlike Eq. 25, Eq. 26 with $\omega = 0.75$ allows the agent to reach goal location (0,0) without needing to extend the trial duration to 3600 seconds.

A.6 Neural implementation of symbolic motor controller

The computations performed by the symbolic motor controller, namely vector subtraction and suppression of motor controller activity, can be learned by a neural network. The neural architecture takes the goal coordinates $g(t)$, confidence function $\varsigma(t)$ and agent's position coordinates $p(t)$ as input $u(t)$, has two layers of 1024 ReLU activated hidden units $h(t)$ and 40 output units with a softmax layer (Fig. 4). The dataset for supervised learning comprised of 923,521 goal, position, confidence function combinations spanning the dimensions of the square maze $x = (0.8m, 0.8m)$ as inputs and the gated symbolic motor controller output ϕ^{mc} as the target. The loss function to be minimised is the mean squared error between the output of the neural ϕ^{nmc} and symbolic ϕ^{mc} and motor controllers.

$$\nabla L = \sum \left(\phi^{mc}(t) - \phi^{nmc}(t)\right)^2 \tag{27}$$

The synaptic weights W^{hidnmc}, W^{outnmc} of the neural network were trained by backpropagation using the Adam optimiser with a learning rate of 0.001 for 20 epochs with $\omega = 0.75$. Such pretraining of neural network to achieve specific computations can be attributed to evolution or learning during development.

This pretrained neural motor controller was integrated with the Reservoir agent to create a fully neural implementation of the Symbolic agent, offering a complete biologically plausible solution to one-shot learning of single displaced and multiple paired associations tasks. Both Reservoir agents were able to replicate the results demonstrated by the Symbolic agent without requiring additional symbolic computations such as suppressing the motor controller activity using $\varepsilon(t)$, instead the pretrained neural motor controller was able to perform implicit gating of activity.

A.7 Symbolic memory & self-position weights

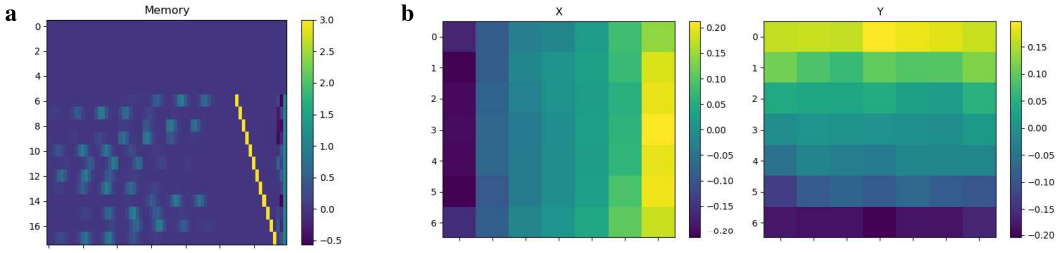

Figure 5: **Episodic memory bank and self-position network weights.** a) 2D episodic memory bank with each row storing 49 place cell activity, cue vector of size 18, 2D goal coordinates and confidence function value of 1 if the target was reached. Example Cue 7 to Cue 18 information are stored in the memory bank. b) Synaptic weights of self-position coordinate network to estimate an agent's 2D coordinate in the square maze. Weights converge to a similar form as in Foster, Morris & Dayan (2000) [14] but for a square maze.

A.8 12NPA trajectory

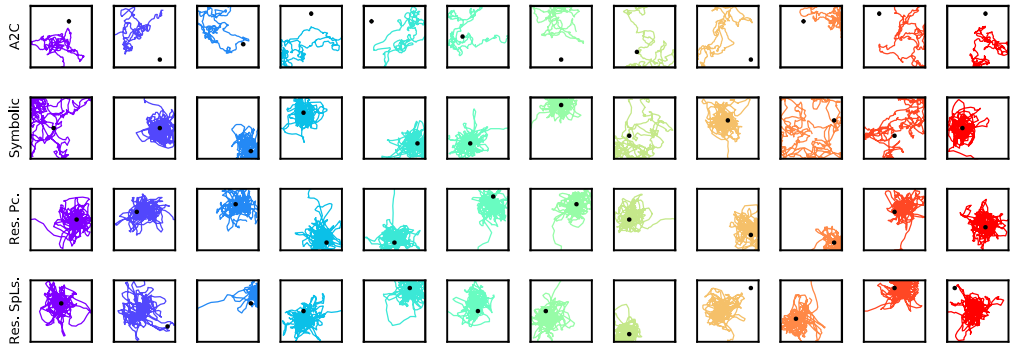

Figure 6: **Example trajectory of agents solving 12 new paired associations (12NPA).** Circles and squares without border indicate start and end positions respectively. A2C agent navigates to five out of 12 locations but achieves low visit ratio at each paired associate. Comparatively, the Symbolic and Reservoir agents spend most of the time at the correct cue-location pairs during each cued probe trial.