An Empirical Evaluation of Neural Network Cognitive Models in Replicating Stochastic Action Switching

Jefri Thimoathi Johnson[1,*], **Michael Condon**[1,*], **Santhosh Raj Srinivasan**[1,*] **and William Goldsmith**[1,*]

[1]*School of Mathematical Sciences, University of Nottingham, Nottingham, NG7 2RD, UK*
[*]*These authors contributed equally to this work*

Abstract

Adaptiveness in decision-making, especially in a dynamic environment, requires balancing exploitation and exploration, where changing ones preference too quickly or slowly can mean missing out on resources. We expand upon a previous study wherein mouse reversal learning in a non-stationary two armed bandit task was simulated with a variety of cognitive models, thus revealing the randomness in action switching and the deviation from optimal Bayesian inference. Furthermore, computational models such as logistic regression, reinforcement learning, and sticky Bayesian inference are used in this model to characterize mouse behaviour, capturing the history-dependent action value and the tendency of the mouse to repeat actions. To gain a deeper understanding of the underlying neural mechanisms. We have integrated a number of recurrent neural network architectures into the existing behavioural data, starting with small gated recurrent units, and building these to switching RNNs and disentangled RNNs. Switching GRU networks take advantage of the discrete state space to create learned nonlinearities with few parameters. DisRNNs offer a structured latent space, where the factors influencing decision-making, such as action sequence and reward history, are learned from data, thereby avoiding researcher subjectivity in identifying the critical factors driving action switching and providing a more understandable model compared to traditional RNNs. This approach helps in bridging the gap between computational models and neural substrates underlying the flexibility in decision-making.

1. Introduction

Beron et al. (2022) investigated how mice adapt to changing rewards in a probabilistic reversal learning task. Mice chose between two side ports, with reward probabilities that reversed unpredictably. The study used computational models to explain switching behaviour, including Q-learning, logistic regression, and sticky Bayesian inference. While mouse choices often deviated from ideal Bayesian strategies, the results showed that simpler models based on recent history could still predict behaviour effectively.

In this project, we re-evaluated those models using the original behavioural dataset. We also introduced modern recurrent neural network architectures to test whether they could capture switching behaviour more accurately. Our goal was to compare model predictions against real mouse data and identify which models best account for adaptive decision-making in this task.

2. Models

2.1. Logistic regression

Logistic regression is a model that can be used to predict how likely an agent is to make a certain choice based on past experiences. It works by combining several input features, such as recent actions or outcomes, and weighting each according to its value. It uses a model where i refers to the type of information being used, such as whether the agent received a reward or which side it chose, and t refers to the current trial. Each of these features, $x_{i,t}$, is given a weight w_i that reflects how strongly it should influence the decision. The model adds up all of these weighted features to produce a single value. This value is then passed through a sigmoid function, $\sigma(z) = \frac{1}{1+e^{-z}}$, which turns it into a probability between 0 and 1. This final number represents how likely the agent is to make a particular choice on the following timestep (Liang et al. 1989).

$$P(\text{chosen action}_t) = \sigma\left(\sum_i w_i \cdot x_{i,t}\right) \tag{1}$$

Beron et al. 2022 modified this model to better capture how mice based their decisions on recent reward history. Instead of using general features $x_{i,t}$ with weights w_i, they replaced them with interaction terms $\bar{c}_{t-i}r_{t-i}$, where c_{t-i} is the choice made, $\bar{c}_{t-i} = 2c_{t-i} - 1$ is another encoding of choice and r_{t-i} is the reward received on trial $t - i$. Each of these interaction terms was assigned a weight β_i to estimate how strongly a rewarded choice influenced the next decision. Furthermore, a new term $\alpha\bar{c}_t$ was introduced to capture a general tendency to repeat the most recent action, independent of reward. These variables are linearly combined. Each past rewarded choice $\bar{c}_{t-i}r_{t-i}$ is multiplied by its corresponding weight β_i, the most recent choice $\bar{c}_t$ is multiplied by α, and all terms are summed together to produce the predictor or log odds, ψ_{t+1}:

$$\psi_{t+1} = \alpha c_t + \sum_{i=0}^{5} \beta_i \bar{c}_{t-i} r_{t-i} \tag{2}$$

which is then passed through the same sigmoid function, $\sigma(\psi_{t+1}) = \frac{1}{1+e^{-\psi_{t+1}}}$, to generate the probability of choosing a particular action on the next trial.

To reduce memory demands and simplify the structure of the model, Beron et al. (2022) replaced the separate choice–reward terms with a single evolving variable ϕ_t. The previous model tracked past events separately

and assigned a different weight to each. The recursive version no longer stores each trial on its own. It builds a single value that gets updated on every trial. This new variable adds the most recent choice and reward, $c_t r_t$, scaled by a learning rate β, to a decayed version of its own previous value. The decay is controlled by a time constant τ, which determines how quickly earlier experiences fade:

$$\phi_t = \beta \bar{c}_t r_t + e^{-1/\tau} \phi_{t-1} \tag{3}$$

This means the model no longer tracks each past trial separately. It gradually builds a summary of reward history by weighting recent rewards more strongly while allowing older ones to fade. The learned parameters for the original logistic regression decreased exponentially, justifying this approximation. The stickiness term $\alpha \bar{c}_t$ stays the same and these are combined into the final prediction:

$$\psi_{t+1} = \alpha \bar{c}_t + \phi_t \tag{4}$$

This is passed through a sigmoid function, $\sigma(\psi_{t+1}) = \frac{1}{1+e^{-\psi_{t+1}}}$, to calculate the probability of choosing the right port.

2.2. Q-learning

Q-learning is a reinforcement learning algorithm where an agent learns from interacting with its environment by estimating how rewarding each possible action is given all following actions are optimal (Sutton et al. 2018). Unlike logistic regression, which models choices based on a fixed summary of past events and is mainly descriptive, Q-learning updates its estimates as new outcomes are observed. Suppose an agent repeatedly chooses between two options, and after each choice, it either receives a reward or it doesn't. The agent remembers how rewarding each action has been in the past using a value called $Q_t(a)$, which represents the expected reward of action a at time t. When the agent chooses an action a_t and observes the outcome, it calculates the difference between what it expected ($Q_t(a_t)$) and what it actually received (r_t). This difference is called the prediction error. The model then updates the Q-value for that action so that future decisions reflect this new experience.

The prediction error ($r_t - Q_t(a_t)$) is also multiplied by a learning rate α, which controls how strongly the agent adjusts its belief. A high learning rate means the agent updates quickly and a low learning rate means it learns more slowly. The model combines these to update the value of the chosen action using the following rule:

$$Q_{t+1}(a_t) = \begin{cases} Q_t(a_t) + \alpha\,(r_t - Q_t(a_t)) & \text{if } i = c_t \\ Q_t(a_t) & \text{if } i \neq c_t \end{cases} \tag{5}$$

However, to better match the behaviour of mice, Beron et al. 2022 modified the standard Q-learning equation by introducing two changes inspired by Ito et al. 2009. Instead of using the original learning rate α, they introduced a new learning rate β_Q. They then added a forgetting component through the term e^{-1/τ_Q}. This allowed the existing Q-value to decay over time, meaning that previously learned values gradually diminished in influence unless reinforced by new rewards . This decay applied to both chosen and unchosen options, but only the chosen option received a reward-based update. In this setup, $Q_{t,i}$ represents the expected value of option i at trial t. r_t is the received reward, and c_t is the chosen action. These changes helped the model account for limited memory and recency effects observed in mouse behaviour.

$$Q_{t+1,i} = \begin{cases} e^{-1/\tau_Q} Q_{t,i} + \beta_Q (r_t - Q_{t,i}) & \text{if } i = c_t \\ e^{-1/\tau_Q} Q_{t,i} & \text{if } i \neq c_t \end{cases} \tag{6}$$

Beron et al. (2022) also tested a simplified variation of Q-learning known as Forgetting Q-learning (F-Q). This version removes the prediction error term from the standard update rule. The original model updates the value of the chosen action based on the difference between the received reward and the expected value. However, in this variant, the prediction error term $r_t - Q_{t,i}$ is removed and the update becomes a direct convex combination of the current value and the observed reward. Meaning instead of adjusting based on error, it moves closer to the new reward by a certain amount The learning rate β_Q is replaced by $1 - e^{-1/\tau_Q}$, where τ_Q is the forgetting time constant that determines how quickly past information fades. This means that newer outcomes are weighted more heavily, while older information decays over time. When the chosen option $i = c_t$, the update rule becomes:

$$Q_{t+1,i} = e^{-1/\tau_Q} Q_{t,i} + \left(1 - e^{-1/\tau_Q}\right) r_t \tag{7}$$

For the unchosen option $i \neq c_t$, the value decays without incorporating a new reward:

$$Q_{t+1,i} = e^{-1/\tau_Q} Q_{t,i} \tag{8}$$

One can combine the Q_t value for both sides into one term $\Delta Q_t = Q_{t,1} - Q_{t,0}$. This gives

$$\Delta Q_{t+1} = e^{-1/\tau_Q} \Delta Q_t + (1 - e^{-1/\tau_Q}) \bar{c}_t r_t \tag{9}$$

where as before $\bar{c}_t = 2c_t - 1$ and to sample the log odds

are passed through the sigmoid function as in logistic regression. The log odds here are of the form:

$$\psi_{t+1} = \alpha \bar{c}_t + \Delta Q_{t+1}/T \tag{10}$$

where α is a learned parameter and T is a temperature parameter. Q-learning is an off-policy method, so it this policy can be chosen arbitrarily as long as it explores the state space. In this case, the log odds are equivalent to the RFLR descriptive model if one sets α the same in both equations, $(1 - e^{-1/\tau_Q}) = \beta$ and $\Delta Q_{t+1}/T = \phi_t$.

3. Model Reproducibility

3.1. Non-Sticky Bayesian Model

To capture belief-based inference in probabilistic reversal learning, Beron et al. 2022 implemented a Bayesian Hidden Markov Model (HMM) agent. The task involves two latent states corresponding to reward contingencies favoring either the left or right option. The agent maintains a belief over these hidden states, updating it trial-by-trial via Bayes' rule using a fixed state transition matrix and the observed action-reward outcomes.

The belief state at trial t, denoted $\mathbf{b}_t = [b_t^0, b_t^1]$, is updated in two steps:

1. **Transition (Prior)**:

$$\mathbf{b}_t^- = T^\top \cdot \mathbf{b}_{t-1} \tag{11}$$

where T is the state transition matrix, typically:

$$T = \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix}$$

2. **Bayesian Update (Posterior)**:

$$b_t^s = \frac{P(r_t \mid a_t, s) \cdot b_t^{-s}}{\sum_{s'} P(r_t \mid a_t, s') \cdot b_t^{-s'}} \tag{12}$$

where $P(r_t \mid a_t, s)$ is the likelihood of the observed reward given the action a_t and latent state s.

The agent chooses actions using a softmax rule:

$$P(a_t = a) = \frac{e^{\beta \cdot b_t^a}}{\sum_{a'} e^{\beta \cdot b_t^{a'}}} \tag{13}$$

where β is the inverse temperature parameter controlling exploration.

This model aligns with the non-sticky Bayesian HMM proposed by Costa et al. 2015, which captured high-reward preferences but overestimated switching behavior. It lacked the inertia observed in real mouse behavior, frequently switching even after consistent rewards.

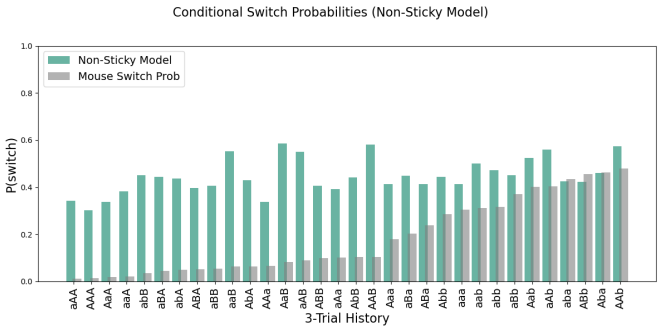

Figure 1. Non-Sticky Bayesian Agent Behavior. Choice probabilities and Pswitch across histories. Moderate switching persists even after stable sequences like `AAA`, reflecting a lack of behavioral inertia.

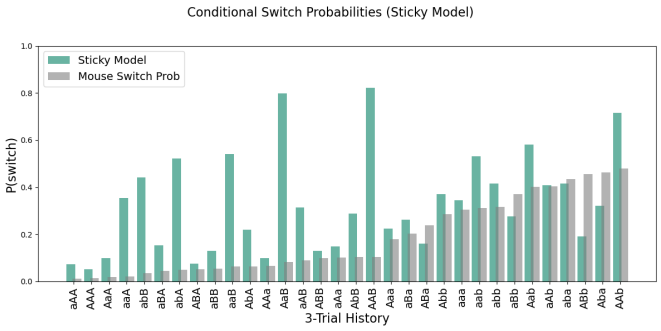

Figure 2. Sticky Bayesian Agent Behavior. Including a stickiness term reduces switching, better capturing empirical patterns of behavioral inertia and habit formation.

3.2. Sticky Bayesian Model

To better match empirical mouse behavior, they extended the model by incorporating a *stickiness* parameter κ, which biases the agent toward repeating the previous action. The belief update remains identical to the non-sticky model, but the action policy is modified as:

$$P(a_t = a) = \frac{e^{\beta \cdot b_t^a + \kappa \cdot \mathbb{I}[a = a_{t-1}]}}{\sum_{a'} e^{\beta \cdot b_t^{a'} + \kappa \cdot \mathbb{I}[a' = a_{t-1}]}} \tag{14}$$

Here $\mathbb{I}[a = a_{t-1}]$ is an indicator function equal to 1 if the agent repeats the previous action. Positive values of κ increase this tendency, capturing action perseveration (Soltani et al. 2010).

This sticky model produced a more biologically plausible switch gradient, especially under stable reward histories. It better aligned with the reduced switching observed in mice following consistent rewards and block reversals.

The sticky model outperformed the non-sticky version in log-likelihood, switch probability alignment, and overall trajectory prediction. It reproduced gradual behavioral adjustment during block transitions and avoided the overly flexible switching seen in the non-sticky agent. These results underscore the role of habit-

ual biases in decision-making and support the inclusion of a stickiness term for accurate behavioral modeling. This is consistent with evidence for action repetition tendencies in cortico-basal ganglia circuits (Las et al. 2005; Soltani et al. 2010).

3.3. Q-Learning Model

Beron et al. 2022 investigated how the mice had to choose between two ports with varying rewards that switched over time, requiring a balance in exploration and exploitation. It was found that the logistic regression model, which clearly explained the choice behaviour in mice, closely resembled Q-learning. the Q-learning model includes parameters such as learning rate (α) and stickiness (β), which modulate how the mice update the action values and repeat previous choices. The Action values represent the expected reward offered for each port.

The study compared three levels of performance, starting with the theoretically optimal Bayesian agent achieving 74% rewards, 70% with actual mouse behaviour, and 69% rewards with the Q-learning model. The Q-learning model achieved near-maximum reward rates, compared to the actual performance of the mice and explained the probabilistic action switching and choice behaviour (Figure 5 and Figure 6). Mice were also sensitive to the non-stationary rewards by adjusting their behaviours. Using the simple form of the logistic regression weights, the model recursively updates a single state estimate, capturing the choice and switching behaviour and generalizing the environmental parameters by reducing the expected rewards. The study also explained the history dependence of behaviour using a unique combination of choice-reward sequences obtained computationally from the preceding trials. The policy implicitly guides the future choices across varying latent states.

3.4. Forgetting Q-Learning

While Beron et al. 2022 primarily focused on standard Q-learning, they have included the concept of "forgetting" to explain the dynamic nature of tasks and their neural mechanisms inspired by Ito et al. 2009. Forgetting or decay in Q-learning reduced the value of past experiences with time, thereby allowing the agent to prioritize more recent information. This concept is particularly relevant in dynamic environments where rewards change frequently. Including the forgetting factor (λ) in the rules of Q-learning, how the action values are updated. Instead of depending completely on the immediate reward and the future reward, the agent also discounts the past values, preventing the agent from being fixated on the past experiences that are no longer relevant. The Q-learning update rule with forgetting is:

$$Q_{t+1}(a_t) = (1-\lambda)Q_t(a_t) + \alpha\,(r_t - Q_t(a_t)) \qquad (15)$$

The study explained the mathematical correspondence between logistic regression and Q-learning models with specific parameter constraints. Even though the predictions made by the models are the same, there are differences in their underlying assumptions. If the weights of logistic regression, which influence the past actions and outcomes, are subject to decay, this would effectively mimic the forgetting process in Q-learning. Especially, the neural mechanisms of synaptic plasticity, such as Long-Term Depression (LTD) could potentially implement this decay (Gerstner et al. 1996). Additionally, the stickiness parameter (β), capturing the tendency to repeat previous actions, can also be modulated by forgetting. In cases of stickiness parameter reducing over time, the agent becomes more flexible and adaptable to the quickly changing reward probabilities. The prefrontal cortex and basal ganglia, involving the neu-

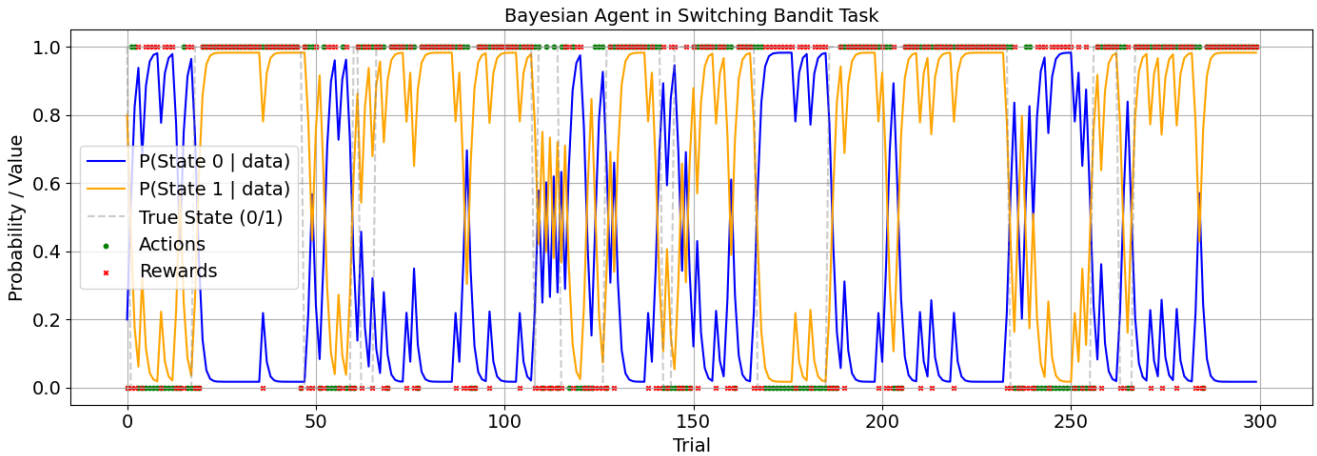

Figure 3. Switching Probability Comparison. The Non-Sticky Bayesian model exhibits higher switching probability compared to the Sticky Bayesian model, due to the absence of behavioral inertia.

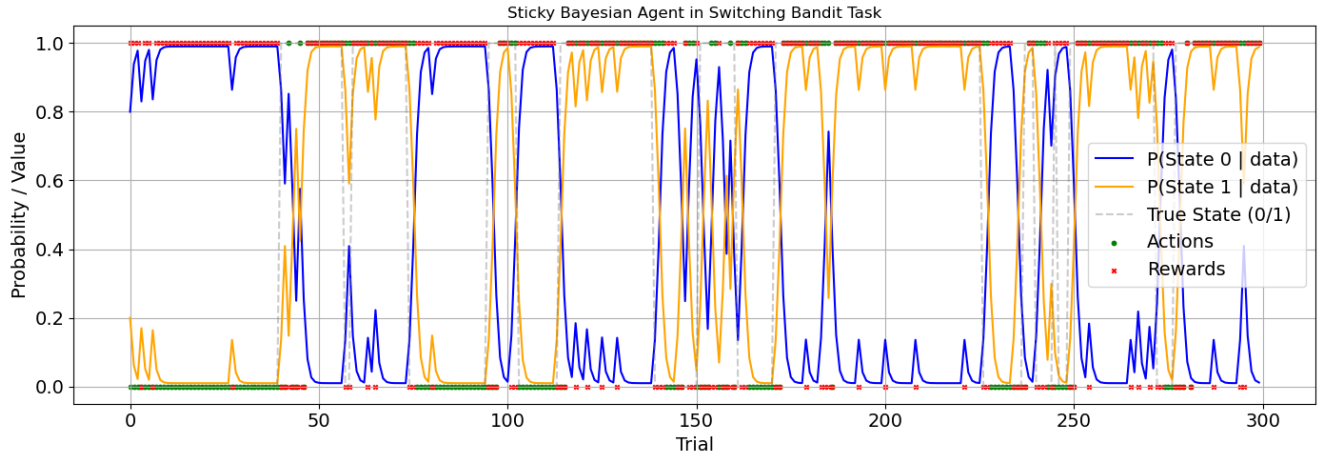

Figure 4. Sticky Agent Performance. Behavior of the sticky agent across blocks. More stable action choices are observed after rewarded histories.

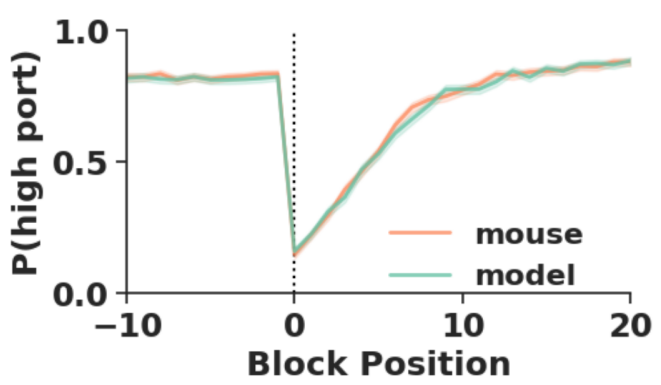

Figure 5. Adaptation of choice probability across trials, Q-learning value updates. Gradual increase in selection of high port by mice after block intervals. The slow convergence shows that the learning rate (α) balancing the recent evidence with past experience.

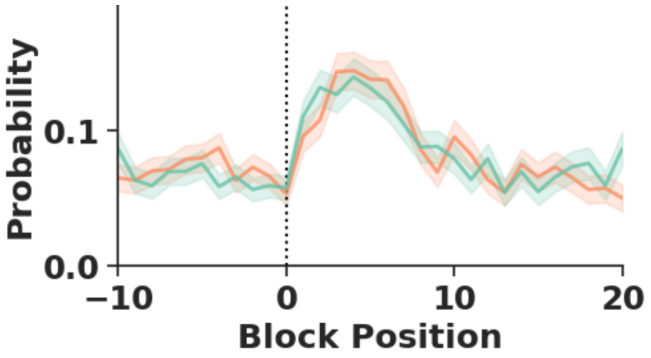

Figure 6. Exploration of stochastic action switching in Q-learning. Mice exhibited low levels in switching but showed increase in exploration near block transitions, similar to the ϵ greedy policy in Q-learning. This asymmetry between the low switches (rewarded) and high switches (unrewarded) reflects the dependent exploration.

ral circuits, which are known to be involved in action selection and reinforcement learning, can potentially mediate the modulation (Yin et al. 2006).

3.5. Logistic Regression

Beron et al. 2022 used Logistic Regression as a descriptive model to capture mouse behaviour. To capture nonlinear interactions between the two input variables, choice and reward, they performed feature engineering to encode the choice and the choice reward interaction as $\bar{c}_t := 2\bar{c}_t - 1$ and $\bar{c}_t r_t$ respectively. Through cross validation, they determined that the only necessary inputs were a history length of 1 for $\bar{c}_t$, and a history length of 5 for $\bar{c}_t r_t$. This was fit by minimising the cross entropy loss which is equivalent to maximum likelihood estimation in the Beron paper. We also use L2 regularisation.

$$\alpha, \beta_1 \cdots \beta_5 = \underset{\alpha,\beta_1\cdots\beta_5}{\arg\min}\, L_{\text{cross entropy}}(P, \hat{P}) + \gamma ||\alpha, \beta_1 \cdots \beta_5||^2 \tag{16}$$

We use Jax and Haiku in Python 3.13 to recreate LR and RFLR in order to make it compatible with further descriptive models we introduce. This logistic regression model does not maintain any internal state, so the only long term dependency is on the last 5 values of the choice reward interaction. Storing the last 5 outcomes and their order in memory is quite unrealistic for a mouse, so to simplify the cognitive requirements, Beron et al. proposed approximating the sum over the previous 5 choice reward interactions with an infinite sum. They found that the β_i weights approximated exponential decay.

Although this descriptive model performs quite well (Table 1) and requires only three parameters, it is not very flexible and some other nonlinearity may provide a better prediction. To explore a wider space of potential cognitive models, we consider a number of recent advances in deep learning for cognitive models.

4. Deep Learning Models

4.1. Gated Recurrent Units

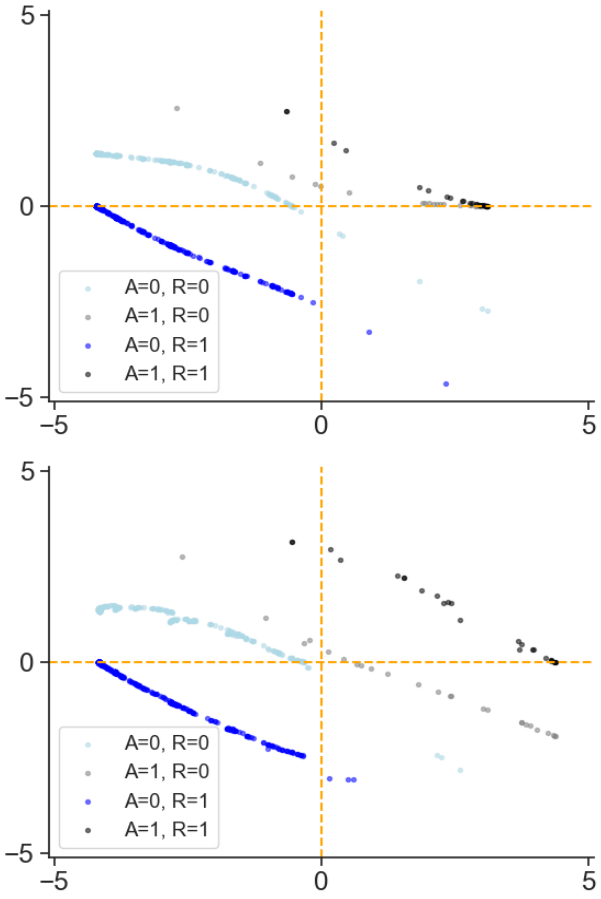

Figure 7. Discrete dynamical system phase plot for (a) the 1 hidden unit GRU and (b) the 2 hidden unit GRU. Note the asymmetry between the left and right choices. With added nonlinearity, there is more symmetry. For the unrewarded right port choice, the fixed point is much further to the right in the single hidden unit plot.

In attempting to replicate the switching behaviour in the mice, Beron et al. 2022 obtained essentially the same formula from the recursive formulation of logistic regression (their descriptive algorithm) and from forgetting Q-learning. Both of these models decay their previous hidden state ϕ_{t-1} exponentially and add a linear function of choice $\bar{c}_t = 2c_t - 1$ a multiple of the choice and the reward $\bar{c}_t r_t$. This feature engineering adds model complexity despite not increasing the number of parameters. In Q-learning, the exact equation for action value, Q, is justified theoretically by the Bellman equation, but no theoretical justification is provided for its use in RFLR and LR in the Beron paper. Since handcrafted models are prone to researcher subjectivity and bias, we seek a balance between interpretability and accuracy with three artificial recurrent neural network techniques. First, we use the usual gated recurrent unit (GRU) architecture, then embellish this with a recent switching modification called switching GRU (SGRU) Chung et al. 2014; Ji-An et al. 2023.

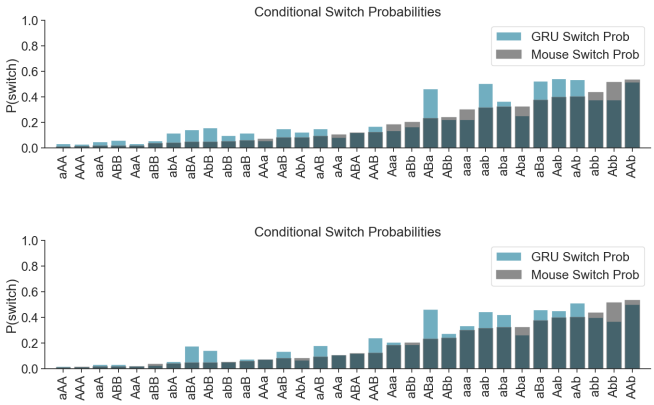

Figure 8. Conditional probabilities of switching for 32 action reward histories of length 3 for GRU vs Mouse. There is a noticeable difference in accuracy between the 1 hidden unit GRU probabilities on top and the 2 unit on the bottom. The latter more accurately reproduces switching.

A GRU network is a type of neural network that operates on sequential data. It maintains a hidden state h_t and at each time step decides whether or not to update it depending on observations x_t. The decision about how to update the hidden state is made by two gating variables, the update gate z_t and the reset gate r_t. These gates are vectors with values between 0 and 1. They are single layer neural networks with a sigmoid activation function σ, weight matrices W_z and W_r and bias terms b_{xz}, b_{hz}, b_{xr} and b_{hr}.

$$z_t^j = \sigma(W_z x_t + b_{xz} + U_z h_{t-1} + b_{hz})^j \quad (17)$$

$$r_t^j = \sigma(W_r x_t + b_{xr} + U_r h_{t-1} + b_{hr})^j \quad (18)$$

First, a new candidate hidden state is constructed with a typical single layer perceptron. This is a linear combination of the observation and the reset gate times the previous hidden state. The reset gate works by multiplying element-wise ($\odot$) each value of the previous hidden state. This is passed through another single layer neural network with weight matrices W and U. The bias terms are b_{xh} and b_{hh}.

$$\tilde{h}_t^j = \tanh(W x_t + b_{xh} + U(r_t \odot h_{t-1}) + b_{hh})^j \quad (19)$$

Finally, the new hidden state is constructed from a linear interpolation of the previous hidden state and the candidate one.

$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \tilde{h}_t^j \quad (20)$$

Since the dimensionality of the hidden state corresponds to the possible memory length, we keep this very low in our experimentation. We experiment with

1 and 2 hidden variables.

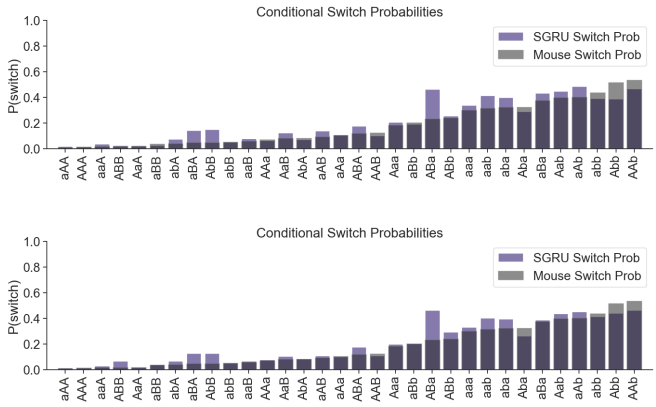

Figure 9. Conditional probabilities of switching for 32 action reward histories of length 3 for SGRU vs Mouse. As shown in the Beron et al. 2022 paper, the symmetry in choice is such that left and right can be combined. In this plot the first choice is always a for unrewarded or A for rewarded regardless of which particular side. The top plot is for the 1 hidden unit SGRU and the bottom one is for the 2 unit variant. The latter is slightly more accurate compared to the non parametric model in grey.

In the two armed bandit task, there are only 4 possible observations at any timestep. In order to represent the discreteness of these observations, Ji-An et al. 2023 propose using a different set of GRU parameters for each possible observation. This SGRU architecture differs from the GRU architecture in two ways. The first is that W_r, W_z and W are all zero. The second way is that a separate parameter set is used for each observation. Intuitively, this approach makes sense, a continuous function of the observations is not required because the only observable values are discrete. A rewarded left port choice, an unrewarded left port choice, a rewarded right port choice and an unrewarded right port choice. We allow one and two latent variables to better interpret the update rules.

4.2. Disentangled RNNs

We also utilise Disentangled RNNs (DisRNNs), another interpretable deep learning approach for the two armed bandit task by Miller et al. 2023. Disentangled RNNs are closely related the the GRU concept, but with information bottlenecks to encourage interpretability. An information bottleneck is simply a normally distributed random variable whose mean is the input multiplied by a learned parameter m and whose variance is a learned parameter σ. For example, the result of passing x through a bottleneck is

$$\text{bottleneck(x)} := \mathcal{N}(mx, \sigma_b) \tag{21}$$

To update its hidden state, a disentangled RNN applies

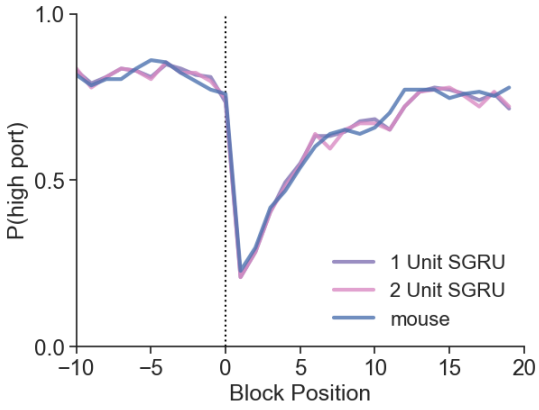

Figure 10. The probability of choosing the target port following a change in reward contingencies. This is slightly underestimated by the SGRU model compared to the mouse.

a bottleneck to the observations and hidden state before they are passed into the update MLP. This update MLP determines update rule for the next hidden state. To do this, it proposes a candidate hidden state $\tilde{h}_t$ and a gating variable z_t.

$$\tilde{h}_t, z_t = MLP_u(\text{bottleneck}(h_{t-1}, o_t)) \tag{22}$$

The gating variable is used to linearly interpolate between the candidate hidden state and the previous hidden state h_{t-1}. The output from this linear combination is passed through another bottleneck before determining the hidden state h_t in the next timestep.

$$h_t^i = \text{bottleneck}((1 - z_t^i)h_{t-1}^i + z_t^i \tilde{h}_t^i) \tag{23}$$

The disentangled recurrent neural network outputs log-odds corresponding to a probability of a left or right choice on the next timestep. This comes from a MLP function of the current hidden state. By passing this through a sigmoid function, one retrieves the normalised probability of a right choice. The total loss for the output is the sum of the cross entropy for the prediction and the Kulback-Leiber divergence D_{KL} between each bottleneck and the unit normal distribution.

$$L_{\text{bottleneck}}(b) = \sum_t D_{KL}(\mathcal{N}(m_b x_t, \sigma_b) || \mathcal{N}(0,1)) \tag{24}$$

An information bottleneck whose outputs are distributed according to the unit normal would have the minimal KL divergence of 0.

$$L_{\text{total}} = L_{\text{cross-entropy}}(\sigma(\hat{y}, y)) + \sum_b L_{\text{bottleneck}}(b) \tag{25}$$

At the start of training, the bottlenecks are open, with approximately 0 variance. Over the course of training, these tend toward 1 such that the total loss is minimised. This is akin to regularisation, but the mean value of the

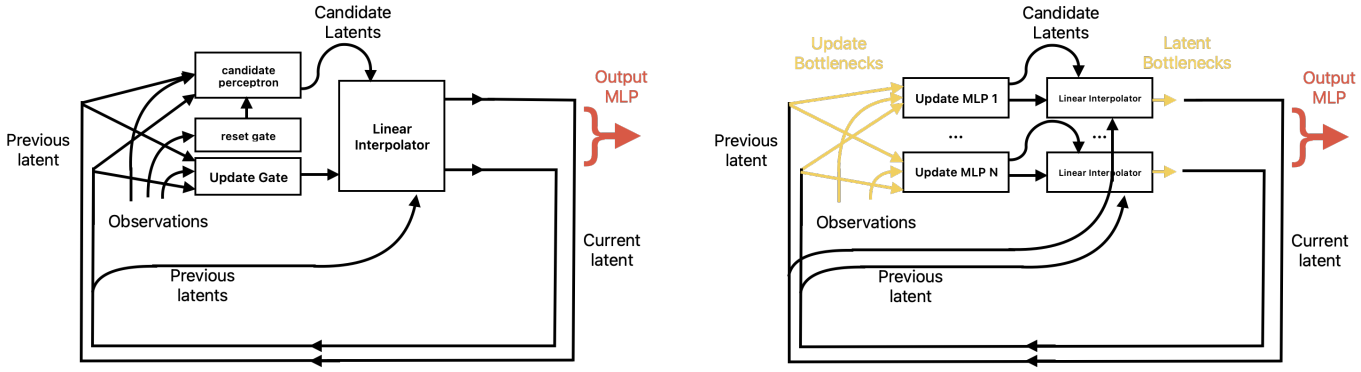

Figure 11. Diagrams for the usual GRU architecture on the left and the disentangled RNNs on the right. The disentangled RNNs do not have a reset gate, but they do have a separate MLP for each latent variable. This update MLP outputs both the candidate latent and the update gate variable. In the GRU case, there is a perceptron for the reset gate, update gate and the candidate latents. Each of these operate on all latents and all observations, outputting gating variables and candidates for all latents.

bottleneck is not necessarily changed during training.

5. Experiments

We used the disentangled RNN implementation provided by Google DeepMind on GitHub (Miller et al. 2023), and adapted it for the two armed bandit data provided by Beron et al. 2022. In order to easily compare the other models, we also implemented the switching GRU and usual GRU architectures, logistic regression and RFLR with Haiku and Jax in Python 3.13. All models were trained on the same 70% of sessions and evaluated on the remaining 30%. All of these models were trained for 10,000 steps to minimise negative log likelihood or cross entropy on batches of 64 sessions at a time. For LR, RFLR, GRU and SGRU models, we used L2 regularisation, but for the disentangled RNNs, we used the bottleneck loss $L_{\text{bottleneck}}$ instead. We trained the disentangled RNN with 4 values of β from 10^{-3} up to 3×10^{-2}. This took about 1 hour each on a 2023 Apple M3 CPU with 8GB of RAM.

Disentangled RNN is the most parameterised model we use. These models were comprised of 5 multi layer perceptrons of depth 3 with 5 units per layer in the update network, and an output network of depth 2 with 2 units per layer. Each MLP updates a latent variable after passing it through an information bottleneck. The bottlenecks simply add gaussian noise to the MLP output. At the start of training, the bottleneck is open with zero variance, but due to an information penalty β, they are closed and only the most necessary bottlenecks remain open. Furthermore, there is also a bottleneck on information entering the update network. These restrictions allow the update rules to be highly nonlinear, but remain interpretable due to the low number of latents and limited interactions between them.

The GRU and switching GRU (SGRU) were also trained for the same number of steps, but took about 3 minute to train. This is mainly because these networks were much smaller. We use only 1 or 2 hidden units in either case with L2 regularisation, which caused a reduced number of effective parameters. We evaluate these models with the log likelihood on the validation set, and validate their ability to reproduce switching behaviour via several graphs of behaviour. The log likelihoods for the validation set are summarised in Table 1 These include plots of the probability of switching following length 3 action reward sequences, plots of the probability of choosing the high port after the reward contingencies change and the probability of switching after reward contingencies change. To understand the dynamics we plot the log odds of taking the right port against the change in log odds (e.g. Figure 7).

6. Results

All of the descriptive models performed similarly, but the best log likelihood achieved in our experimentation by interpretable models was -0.195, by both SGRU with 2 hidden units and disRNNs with $\beta = 0.001$ rather than the -0.18 in the Beron paper. This discrepancy is likely due to a difference between our train test split and theirs. We observed that for all models, thompson or stochastic sampling reproduced switching dynamics better than greedy sampling as in the Beron paper.

We trained the Disentangled RNN architecture with 4 values of β. We found that a lower penalty term β lead to a better log likelihood with the best LL=-0.198 for $\beta = 0.001$. With this value, the disRNN maintained 2 latent variables. The inputs to the update rule for the latent variables can be observed in the heat map (top right of Figure 14), with the first depending on reward, choice and itself and the second latent variable depending on solely on choice. The update rules for

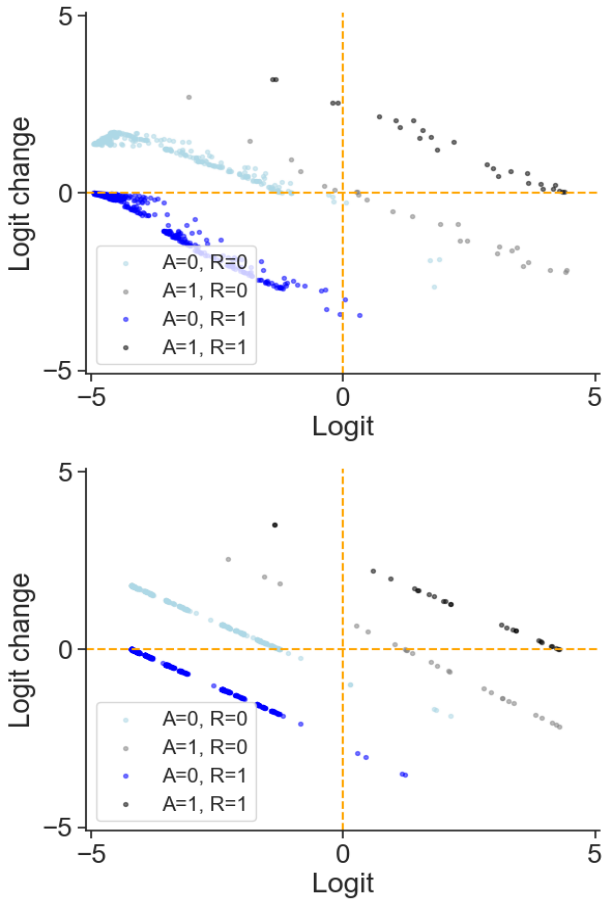

Figure 12. Discrete dynamical system phase plot for the disRNN with $\beta = 0.001$ on top and $\beta = 0.03$ on the bottom. As can be seen in Figure 14, two latent variables are active in the first case, but only one latent variable is active in the second case.

these latents can be seen in the corresponding lower plots. For the first latent variable, the rules appear nonlinear. Only extreme latent values are regulated toward zero when a state is rewarded, but for the unrewarded case, nearly all latent values are regulated toward zero since the curve crosses unity near zero and has absolute value less than the unity line. The choice dependent latent cannot be interpreted fully from the latent plots since the correspondence between latent and log-odds is not obvious. Instead, since the RNNs we consider are Markovian, we can consider them as a discrete dynamical system. In Figure 12, each reward condition from a single session is plotted as a point in phase space for the high penalty, unsophisticated model on the bottom and the high penalty sophisticated model on the top. The more sophisticated model has both unrewarded conditions with approximately zero change for logits with zero magnitude, however for the unrewarded conditions, there is a curve for larger logits. This curve means that if the model gives a large or confident logit value, the update size does not increase linearly. This contributes to slight perseveration. For the simpler

Model	Hyperparam	Metric	# Params
LR	–	-0.198	3
RFLR	–	-0.198	6
DisRNN	0.03	-0.202	664
DisRNN	0.01	-0.200	664
DisRNN	0.003	-0.198	664
DisRNN	0.001	-0.195	664
GRU	1	-0.215	14
GRU	2	-0.199	33
SGRU	1	-0.198	48
SGRU	2	-0.195	122

Table 1. Comparison of descriptive model performance and parameter counts on 30% held out data. The hyperparameter for DisRNNs is β, the penalty term. For SGRU and GRU, it determined the number of hidden units. The metric in all cases is log likelihood. LR and RFLR are models from the Beron paper, but these values are from our implementation. DisRNN code is modified from Miller et al. 2023. GRU and SGRU are implemented fully by us based on Ji-An et al. 2023.

model, this curve is not present. In the simpler model, it is also the case that both unrewarded conditions do not have an identical fixed point at zero unlike the sophisticated model. The fact that these points do not form a straight line through zero means that the unrewarded simple model has a region around zero where its confidence in its chosen value increases despite being unrewarded (its negative values decrease and its positive values increase). This is an inappropriate form of choice perseveration. Finally, one can see that the more sophisticated DisRNN predicts switching more accurately based on Figure 13.

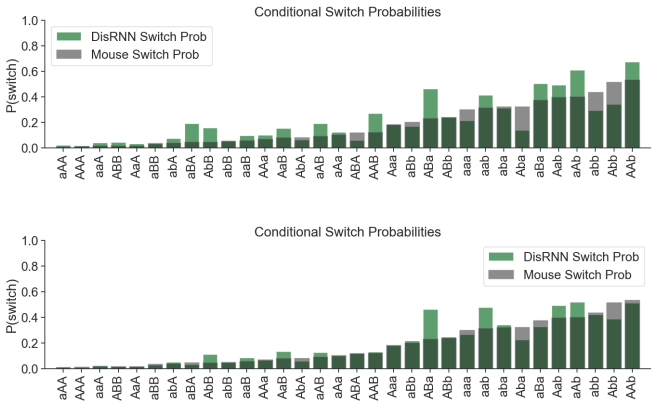

Figure 13. Conditional probabilities of switching for 32 action reward histories of length 3 for DisRNN vs Mouse. There is a noticeable difference in accuracy between the $\beta = 0.03$ probabilities on top and the $\beta = 0.001$ on the bottom. The latter more accurately reproduces switching.

Using the usual GRU architecture with only two

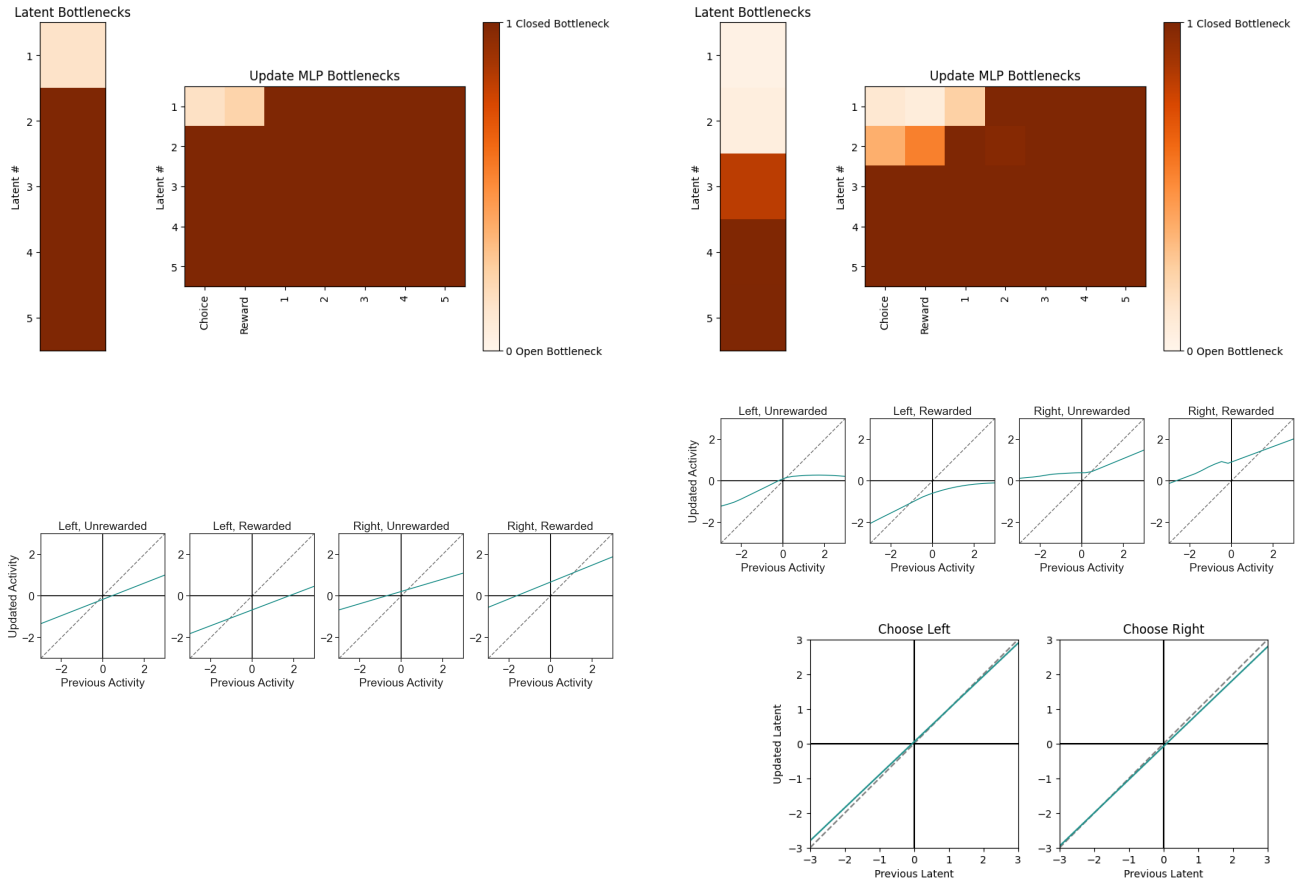

Figure 14. (Top) A heatmap of the bottleneck variance for the disentangled RNNs trained with $\beta = 0.03$ left and $\beta = 0.001$ right, along with the latent update rule plots below their respective bottleneck plots. This plot shows that

gated recurrent units led to quite good performance (LL=-0.199), but we saw much worse performance with only one hidden unit (LL=-0.215). In terms of switching performance, sampling at a rate proportional to the predicted probability was more accurate than greedy sampling. Both 1 unit and 2 unit GRU networks slightly overestimate the probability of taking the high reward port directly after a block transition, but behave more like the mouse a few trials later.

By analysing the phase plot of the GRU networks, it becomes apparent where the GRU falls short compared to the disentangled RNNs. For the 1 unit GRU, the log odds evolve as expected, coinciding with the disentangled RNN when choosing the left port, but when the right port is chosen, the log odds never decrease even when unrewarded (Figure 7). Instead, they increase or stay the same. This behaviour is like that observed with logistic regression by Beron, but here it is not symmetric. In the 2 unit GRU, the log odds dynamics are more like the disentangled RNN in that an unrewarded right port choice provides evidence toward the left port, but the curve for higher values is not present. This results in less choice perseveration for unrewarded right choices.

Both of the 1 and 2 unit switching GRUs perform better than both of the 1 and 2 unit GRU networks. (-0.198 for the 1 unit SGRU and -0.195 for the 2 unit SGRU). They slightly underestimate the probability of the high reward port directly after a block transition (Figure 10), but their conditional switching probabilities based on history is quite accurate in Figure 9. Even with just one hidden unit, the non linearity for unrewarded choices is

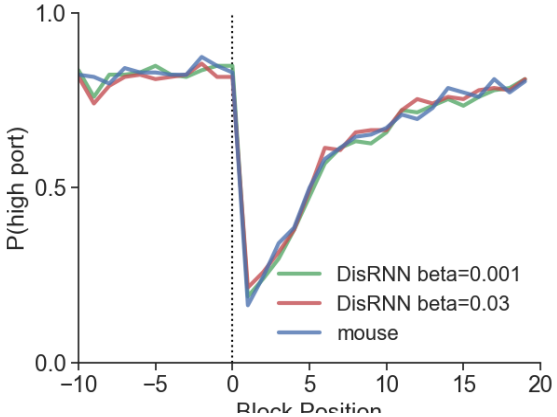

Figure 15. Probability of choosing the target port following a change in reward contingency. The higher β value leads to overestimating the probability of choosing the high port.

replicated and the phase plot is symmetrical (Figure 16). The only observable difference between the dynamics of the 1 and 2 unit SGRU is that for the 2 unit networks, the plotted points do not exist only along one dimensional curves. This is simply due to the extra dimension of the second hidden unit. Since the switching GRU learns a separate parameter set for each observation, it is worth noting how symmetrical the dynamics are between a left and right choice. One could modify the switching GRU so that one parameter set is learned for unrewarded observations and another is learned for rewarded observations, mirroring between sides. This would almost halve the total number of parameters. Under this assumption, the SGRU would perform better than the GRU network with few further parameters. For further work on two armed bandit problems, we find this approach promising. Furthermore, we suspect there is something missing from the F-Q learning and RFLR approaches taken by Beron et al. 2022. The RNN techniques we have mentioned are strictly Markovian, like both of the reduced Beron models, but perform slightly better on the held out data with minimal parametrisation.

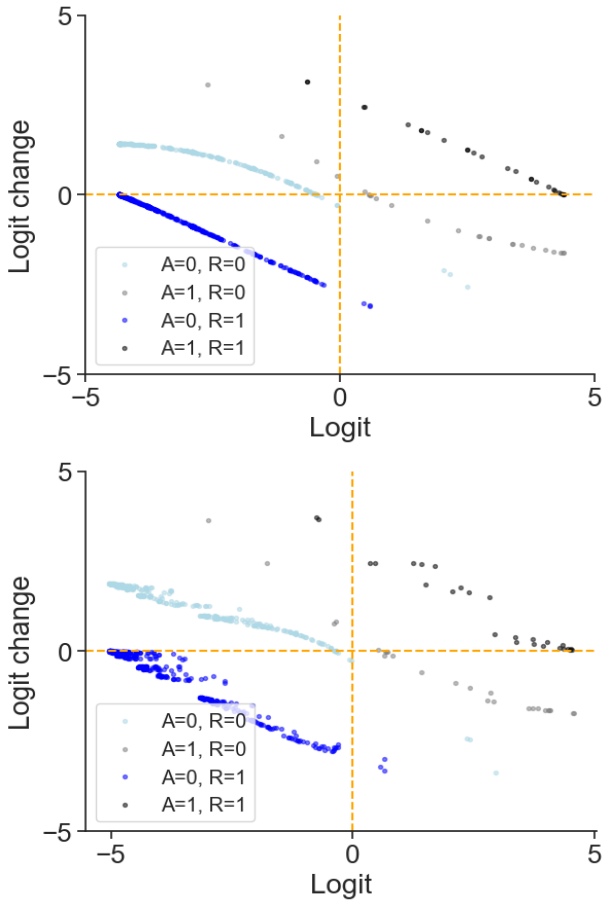

Figure 16. Discrete dynamical system phase plot for (a) the 1 hidden unit SGRU and (b) the 2 hidden unit SGRU.

7. Discussion

The Bayesian Hidden Markov Model: originally proposed by Beron et al., systematically overestimated switching behaviour, particularly after extended reward sequences. In our replication, the non-sticky model predicted switch probabilities around 0.4 even for stable histories like "AAA," where mice switched below 0.1 (figure 1). Adding a stickiness term improved this by suppressing switching after consistent rewards, lowering the model's switch probability to between 0.05 and 0.08 (Figure 2). This brought it closer to the empirical range of 0.02 to 0.05. However, even with stickiness, the model continued to overestimate switching after rewarded switches, assigning probabilities near 0.6 for "aaB". These results indicate that adding stickiness improved model fit but failed to provide the flexibility needed for outcome-dependent persistence.

The forgetting Q-learning model: offered a reinforcement-based alternative. It successfully captured reward-driven adaptationin response to changes in the reward environment. The model's high-port choice probability fell from 0.85 to between 0.05 and 0.1 (figure 5). After a reversal and gradually recovered to around 0.9 by trial 20. This closely matched the mouses behaviour. Switching probabilities also rose (figure 6) appropriately from 0.05 to 0.15. These results support the idea that decaying value estimates can explain learning and exploration without tracking hidden states. However, the model used the same update rule across all outcomes. This prevented it from adjusting switching behaviour based on the reward context, limiting its ability to reproduce asymmetric patterns like stronger perseverance after long reward runs.

The logistic regression and reduced-form variants confirmed that predictive accuracy does not require complicated memory. The original LR model included five reward–choice interaction terms and one choice bias term and achieved a held-out log-likelihood of –0.198 (Table 1). The reduced model instead used a recursive summary variable ϕ_t and reached similar performance using only six parameters. These models matched the GRU's log-likelihood while using far fewer parameters and this supports the claim that much of mouse switching behaviour can be captured by short-term history-dependent rules. However, the reliance on short-term summary statistics made them less flexible. They could not represent longer dependencies or behavioural shifts that require different update dynamics across conditions.

Our implementations of these models have matched the reported log-likelihoods, switching patterns, and reward adaptation curves of the Beron et al study. It confirms that lightweight models based on recent outcomes

can account for much of the observed behaviour. However, none of these models fully captured asymmetric switching responses or selectively modulated commitment following different reward outcomes. This gap was furtherer explored with additional model attempts.

GRUs: applied a uniform update mechanism to all inputs. The 2-unit GRU reached a log-likelihood of –0.199 (Table 1). However, it failed to reduce commitment after errors. In phase space plots, it continued reinforcing the same action after unrewarded trials, unlike the mice who became more exploratory. This suggests GRUs encode inertia but miss outcome-specific adjustments.

SGRUs: improved on this by assigning different update rules to each observation. This allowed the model to learn to respond differently to rewards and non-rewards. The 2-unit SGRU matched the best observed performance (LL = –0.195) and produced behaviour closely aligned with the mouse data. It reduced switch probability after rewarded trials and increased it after unrewarded ones, as seen in conditional switch plots. The symmetry and targeted modulation of its dynamics were evident in the phase space plots, reflecting more flexible, context-aware learning.

Disentangled RNNs: achieved the same likelihood (LL = –0.195) while offering interpretability through an internal bottleneck. One latent variable tracked reward-dependent behaviour, another tracked choice history. The reward-sensitive latent flattened after errors and ramped up with consistent rewards and this mimicked the animal's shift between exploration and commitment. The DisRNN matched switching rates across histories and successfully tracked port preference changes after reversals. Though it did not outperform the SGRU, it provided insight into the internal structure of behavioural control, offering a new way to interpret learned strategies.

In conclusion: Our replications confirmed that the modelling framework used by Beron et al. (2022) was both valid and effective. Our implementations reproduced their results and demonstrated that simple reinforcement-based and descriptive models can account for the core features of mouse decision-making. By adding recurrent neural network models, we showed that letting the model respond differently to rewards and build its own memory helped it match mouse behaviour more closely and made it easier to understand how decisions were made. These additions did not overturn Beron et al.'s conclusions but rather reinforced and deepened them, indicating that while belief-based inference is not necessary, recurrent architectures can capture more detailed aspects of behavioural flexibility.

References

Ji-An, Li, Marcus K. Benna, and Marcelo G. Mattar (Apr. 13, 2023). *Discovering Cognitive Strategies with Tiny Recurrent Neural Networks*. DOI: 10.1101/2023.04.12.536629. Pre-published.

Beron, Celia C. et al. (Apr. 2022). "Mice exhibit stochastic and efficient action switching during probabilistic decision making". en. In: *Proceedings of the National Academy of Sciences* 119.15, e2113961119. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.2113961119.

Chung, Junyoung et al. (Dec. 11, 2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. DOI: 10.48550/arXiv.1412.3555. arXiv: 1412.3555 [cs]. Pre-published.

Costa, Vincent D et al. (2015). "Reversal learning and dopamine: a Bayesian perspective". In: *Journal of Neuroscience* 35.6. [link], pp. 2407–2416.

Gerstner, Wulfram et al. (Sept. 1996). "A Neuronal Learning Rule for Sub-Millisecond Temporal Coding". In: *Nature* 383.6595, pp. 76–78. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/383076a0.

Ito, Makoto and Kenji Doya (Aug. 2009). "Validation of Decision-Making Models and Analysis of Decision Variables in the Rat Basal Ganglia". en. In: *The Journal of Neuroscience* 29.31, pp. 9861–9874. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.6157-08.2009.

Las, Liora, Edward A. Stern, and Israel Nelken (2005). "Representation of Tone in Fluctuating Maskers in the Ascending Auditory System". In: *Journal of Neuroscience* 25.6, pp. 1503–1513. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.4007-04.2005. eprint: https://www.jneurosci.org/content/25/6/1503.full.pdf.

Liang, Kung-Yee and Scott L Zeger (1989). "A Class of Logistic Regression Models for Multivariate Binary Time Series". In: *Journal of the American Statistical Association* 84.406, pp. 447–451.

Miller, Kevin et al. (2023). "Cognitive model discovery via disentangled RNNs". In: *Advances in Neural Information Processing Systems* 36, pp. 61377–61394.

Soltani, Alireza and Xiao-Jing Wang (2010). "Synaptic computation underlying probabilistic inference". In: *Nature Neuroscience* 13.1. [link], pp. 112–119.

Sutton, Richard S. and Andrew G. Barto (2018). *Reinforcement Learning: An Introduction*. 2nd ed. Cambridge, MA, USA: The MIT Press. ISBN: 9780262039246.

Yin, Henry H. and Barbara J. Knowlton (June 2006). "The Role of the Basal Ganglia in Habit Formation". In: *Nature Reviews Neuroscience* 7.6, pp. 464–476. ISSN: 1471-003X, 1471-0048. DOI: 10.1038/nrn1919.