A APPENDIX

A.1 CLOSED-FORM SOLUTION FOR REWARD RATE IN FIXED CONTEXT

Determining the optimal strategy for action requires estimating $P(s_t|\boldsymbol{x}_{\leq t})$ and thresholding on its value. Because $x_t = 0$ implies $s_t = 0$, it is sufficient to count cues from the last negative observation $x_{t-\tau-1} = 0$, i.e. $P(s_t|\boldsymbol{x}_{\leq t}) = P(s_t|\boldsymbol{x}_{t-\tau:t} = 1)$, where we use the slicing notation $t - \tau : t$ to mean all times t' fulfilling $t - \tau \leq t' \leq t$. The waiting time τ hence forms the central component in the actor's policy.

For brevity, we further define

$$b \equiv P(x_t = 1, s_t = 0|s_{t-1} = 0,\, \theta) = \theta\,(1 - \lambda)$$
$$c \equiv P(x_t = 0, s_t = 0|s_{t-1} = 0,\, \theta) = (1 - \theta)\,(1 - \lambda)$$

Transition after τ observations. In order to evaluate the reward rate in the main text, we need to calculate $P(s_t = 1|\boldsymbol{x}_{\leq t},\, \theta)$. Because an unsafe cue $x_t = 0$ always implies the safe state, it is sufficient to calculate $P_{s=1} \equiv P(s_t = 1|\boldsymbol{x}_{t-\tau:t} = 1,\, \theta)$, i.e. how likely the safe state is after τ suggestive observations, where we introduced $P_{s=1}$ for brevity. To this end, we consider the complement $P_{s=0} = 1 - P_{s=1}$. To evaluate this expression, we formulate a self-consistent relation,

$$P_{s=0} = b^\tau + \sum_{k=0}^{\tau-1} b^k c\, P_{s=0}$$
$$\Rightarrow P_{s=0} = \frac{b^\tau}{1 - \sum_{k=0}^{\tau-1} b^k c}.$$

This equation considers the probability b^τ to have been unsafe the entire time. On top of this, the second sum considers all worlds where k observations $x = 1$ where followed by a faithful negative $x = 0$. By the Markov property of the world, these paths again reweight $P_{s=0}$ itself. Resubstitution finally yields $P_{s=1} = 1 - P_{s=0}$. To cross-check this relation, we simulated it via Monte Carlo sampling of the world.

Expected action time. To arrive at the final expression for the reward rate, the average time to reach τ cues is required, as well. We here extend the calculation presented by Ginsparg (2005) for the case of coin tosses with the latent context in our setting.

Let $\bar{T} \equiv \mathbb{E}\left[T \,|\, \boldsymbol{x}_{t-\tau:t} = 1, \theta\right]$ denote the expected action time, where τ indicates the period of consecutive cues $x_{t-\tau:t} = 1$ preceding any action. Then, we arrive at a self-consistent relation

$$\bar{T}(\tau;\theta) = b^\tau \tau + \sum_{k=0}^{\tau-1} b^k \left(\lambda\,\tau + c\left(\bar{T}(\tau,\theta) + k + 1\right)\right)$$
$$\Rightarrow \bar{T}(\tau;\theta) = \frac{b^\tau \tau + \sum_{k=0}^{\tau-1} b^k \lambda\,\tau + b^k c\,(k+1)}{1 - \sum_{k=0}^{\tau-1} b^k c}.$$

This considers all possible paths to get τ consecutive $x_{t-\tau:t} = 1$: The first term is the probability of getting τ *unsafe* cues in $s_{t-\tau:t} = 0$ rightaway. The sum then considers paths where k cues have been unsafe. From there on, a transition to *safe* happens with probability a, and all subsequent cues in this trial will be guaranteed to be faithfully *safe*. With probability c however, the world does not transition to *safe*. Again because of the Markovian structure, the world statistically indifferent from the initial time. Therefore, agent has to wait for $\bar{T}(\tau;\theta)$ on expectation again, in addition to the cost $k + 1$ that it lasted to get to this point, with $+1$ accounting for the negative cue.

From this, the reward rate becomes

$$\mathbb{E}\left[r_t|\boldsymbol{x}_{\leq t},\ \theta\right](\tau) = \frac{1}{\bar{T}(\tau,\theta)} P(s_t = 1|\boldsymbol{x}_{\leq t},\, \theta) = \frac{1}{\bar{T}(\tau,\ \theta)} P(s_t = 1|\boldsymbol{x}_{t-\tau:t} = 1,\, \theta).$$

The maximizer $\tau^\star$ of this expression defines the optimal policy in a context θ.

A.1.1 LATENT VARIABLES REQUIRE TRACKING JOINT DENSITY

In this section, we show that updating of the marginals of the joint belief cannot be done without holding track of the full joint distribution at the previous timestep. This necessitates using the full joint update equation in the main text.

State

$$
\begin{aligned}
P(s_t|\boldsymbol{x}_{\leq t} = (x_t, \boldsymbol{x}_{<t})) &\propto P(s_t, x_t|\boldsymbol{x}_{<t}) \\
&= \sum_{\theta_t} P(s_t, x_t, \theta_t|\boldsymbol{x}_{<t}) \\
&= \sum_{\theta_t} P(x_t, \theta_t|s_t, \boldsymbol{x}_{<t}) P(s_t|\boldsymbol{x}_{<t}) \\
&= \sum_{\theta_t} P(x_t|s_t, \theta_t) P(\theta_t|s_t, \boldsymbol{x}_{<t}) P(s_t|\boldsymbol{x}_{<t}) \\
\text{def. conditional} &= \sum_{\theta_t} P(x_t|s_t, \theta_t) \frac{P(s_t, \theta_t|\boldsymbol{x}_{<t})}{P(s_t|\boldsymbol{x}_{<t})} P(s_t|\boldsymbol{x}_{<t}). \\
&= \sum_{\theta_t} P(x_t|s_t, \theta_t) P(s_t, \theta_t|\boldsymbol{x}_{<t}) \\
&= \sum_{\theta_t} P(x_t|s_t, \theta_t) \sum_{s'_{t-1}} \boldsymbol{T}_{s_t s'_{t-1}} \sum_{\theta'_{t-1}} \boldsymbol{T}_{\theta_t \theta'_{t-1}} P_{t-1}(s'_{t-1}, \theta'_{t-1}|\boldsymbol{x}_{<t}).
\end{aligned} \tag{7}
$$

Importantly, in the third line, $P(s_t|\theta_t, \boldsymbol{x}_{<t}) \neq P(s_t|\boldsymbol{x}_{<t})$, in general. This can be seen from the fact that high noise will make the safe state more unlikely.

Context This follows completely analogously

$$
\begin{aligned}
P(\theta_t|\boldsymbol{x}_{\leq t} = (x_t, \boldsymbol{x}_{<t})) &\propto P(\theta_t, x_t|\boldsymbol{x}_{<t}) \\
&= \sum_{s_t} P(s_t, x_t, \theta_t|\boldsymbol{x}_{<t}) \\
&= \sum_{s_t} P(s_t, x_t|\theta_t, \boldsymbol{x}_{<t}) P(\theta_t|\boldsymbol{x}_{<t}) \\
&= \sum_{s_t} P(x_t|s_t, \theta_t) P(s_t|\theta_t, \boldsymbol{x}_{<t}) P(\theta_t|\boldsymbol{x}_{<t}) \\
\text{def. conditional} &= \sum_{s_t} P(x_t|s_t, \theta_t) \frac{P(s_t, \theta_t|\boldsymbol{x}_{<t})}{P(\theta_t|\boldsymbol{x}_{<t})} P(\theta_t|\boldsymbol{x}_{<t}) \\
&= \sum_{s_t} P(x_t|s_t, \theta_t) P(s_t, \theta_t|\boldsymbol{x}_{<t}) \\
&= \sum_{s_t} P(x_t|s_t, \theta_t) \sum_{s'_{t-1}} \boldsymbol{T}_{s_t s'_{t-1}} \sum_{\theta'_{t-1}} \boldsymbol{T}_{\theta_t \theta'_{t-1}} P_{t-1}(s'_{t-1}, \theta'_{t-1}|\boldsymbol{x}_{<t}).
\end{aligned} \tag{8}
$$

Note that these relations are completely equivalent to each other, up to a normalization.

In summary, this shows that the update of the marginals necessitates keeping track of the joint priors. Intuitively, this comes about in line 3, where the belief about the *joint* occurrence of s_t, θ_t enters. Note that we can recover these relations by marginalization of the joint update in the main text, for

example for the s_t update

$$\begin{aligned} P(s_t|\boldsymbol{x}_{\leq t}) &= \sum_{\theta_t} P(s_t, \theta_t|\boldsymbol{x}_{\leq t}) \\ &= \sum_{\theta_t} P(x_t|s_t, \theta_t) \sum_{s'_{t-1}} \boldsymbol{T}_{s_t s'_{t-1}} \sum_{\theta'_{t-1}} \boldsymbol{T}_{\theta_t \theta'_{t-1}} P_{t-1}(s'_{t-1}, \theta'_{t-1}|\boldsymbol{x}_{<t}). \end{aligned}$$

where we used the normalization of the transition matrix and the prior in the last two steps. This goes likewise for $P(\theta_t|\boldsymbol{x}_{\leq t})$.

A.1.2 Independent observation generator (naive Bayes)

To analyze the effect of the interaction of the variables, we consider a factorized generator, i.e. $P(x|s, \theta) = P(x|\theta)P(x|s)$. This makes s and θ *non-concurring* causes of the observations, which means that the presence of either increases the chances of $x = 1$. This yields independent update equations, as can be seen by carrying out $\sum_{\theta'_{t-1}}$ and $\sum_{s'_{t-1}}$ respectively in (8) and (7),

$$\begin{aligned} P(s_t|\boldsymbol{x}_{\leq t}) &= \sum_{\theta_t} P(x_t|s_t) \sum_{s'_{t-1}} \boldsymbol{T}_{s_t s'_{t-1}} P_{t-1}(s'_{t-1}|\boldsymbol{x}_{<t}) \\ P(\theta_t|\boldsymbol{x}_{\leq t}) &= \sum_{s_t} P(x_t|s_t) \sum_{\theta'_{t-1}} \boldsymbol{T}_{\theta_t \theta'_{t-1}} P_{t-1}(s'_{t-1}|\boldsymbol{x}_{<t}). \end{aligned} \tag{9}$$

Because of this mismatching observation model, keeping track of the state becomes completely decoupled from context. Similar to the main text, this technically assumes a factorized prior in the distant past.

A.2 Network training

We trained recurrent network models, in PyTorch, using the Actor-Critic framework (Sutton & Barto, 2018) . The Actor and Critic each consisted of an LSTM cell with 48 hidden units. The Critic LSTM was followed by a linear readout to value. The Actor LSTM was followed by a linear readout with a softmax activation function that produced 2 action-probabilities (Action and Inaction). The policy was generated by sampling from the action-space with the action-probability produced by the Actor. At each time step both LSTMs recieved the one-hot encoded observation from the environment, the one-hot encoded previous step action, and the binary-encoded previous step reward. The parameters of all layers were initialized using Pytorch's standard initialization for linear layers: $U(\frac{-1}{\text{input dimensions}}, \frac{1}{\text{input dimensions}})$. LSTM units were initialized to zero at the first step of the first episode.

Critic weights were learned using the Mean Squared Error between the Expected Value at each time steps and the Discounted Future Returns, G_t:

$$\text{Critic loss} = \frac{1}{T} \sum_{t=0}^{T-1} A_t^2$$

where A_t is the difference between the Expected Value and the Discounted Future Returns:

$$A_t = V_t - G_t$$

The sum of discounted future returns G_t was computed in reverse at the end of an episode as

$$G_t = \gamma G_{t+1} + R_t$$

where the discount factor γ was set to 0.995. The final return G_T was manually set to the final predicted value, V_T, to avoid a dependence between G_t and the distance to the end of the episode. Actor weights were learned using the policy gradient algorithm (Sutton & Barto, 2018). A learning rate of 0.0001 in combination with an RMSprop optimizer was used for training both Actor and

Critic. The Critic loss was scaled down by 0.05 relative to the Actor, mirroring the approach in Wang et al. (2018).

Each of the 5000 training episodes consisted of 500 trials under a single θ. After each episode a new θ was sampled from the set $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. Trials consisted of a variable number of time steps. Each trial's duration depended on the stimulus duration, agent's behavior, and ITI duration. At the end of an episode the gradient with respect to the weights was computed with backpropagation through time, for approximately 15,000 steps per episode, and the gradient was detached at the end of an episode.

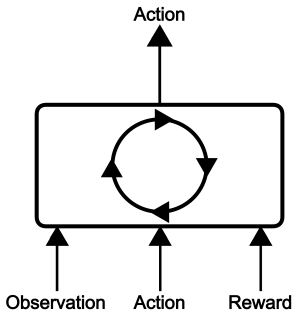

Figure 6: **Graphical representation of the actor's LSTM architecture, inputs and outputs.**

A.2.1 Training dynamics

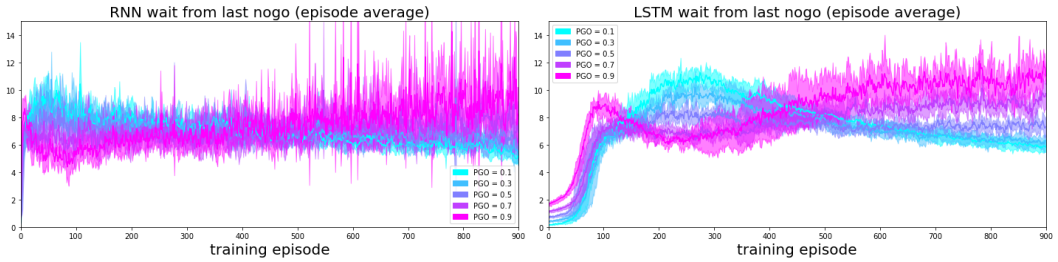

Figure 7: **Training dynamics of 5 RNNs and 5 LSTMs.** Thick lines are means of networks, Error band are standard deviations across networks.

We started out our experiments with training a recurrent neural network on the task. Due to the sparse reward, its dynamics were highly unstable. Figure 7 contrasts the waiting time τ underlying the policy throughout training. As it underlies the optimal reward rate (A.1), this forms a proxy for network performance throughout training. We contrast the plain RNN architecture to the LSTM architecture that we use for the results in the main text. For both architectures we trained 5 networks with identical hyper-parameters except that the learning rate was 0.00001. The purpose of this change was to improve training stability of the Vanilla RNNs. This reveals that the network behavior evolves as training progresses. In a first stage, waiting time increases independent of context from a baseline due to network initialization. In a second stage, low θ delays action, while high θ paradoxically promotes shorter action times. This reflects the association of cues with rewards: The greater frequency of *go* cues in the high noise contexts bring the networks to act sooner. This trend reflects the behavior of the independent theory in the main text. At a last stage, this trend inverts to reflect awareness of the ambiguity of cues, arriving at the final policy.

A.3 Detailed structure of the mouse task

In order to make the task suitable for rodent experiments required a suitable task design on which we detail here. The *unsafe* state duration was sampled from an exponential distribution with a mean of $\lambda^{-1} = 10$ steps and a minimum of 4 (in training) or 1 (in testing). Acting during the stimulus automatically transitioned the state to an interstimulus interval (ITI). The ITI duration was sampled from a uniform duration between 5 and 25 steps. If the agent acted in the safe state, it received a reward of 1. If the agent stayed in the safe state for 35 (in training) or 100 (in testing) steps without

acting, the state automatically transitioned to ITI. The unsafe and ITI stimuli were independently sampled from a Binomial distribution with the observation generator is $P(x = 1|s = 0, \theta) = \theta$. After a *block* of trials θ was sampled uniformly from a list of possible values.

A.3.1 BAYESIAN AGENT

Action selection. In order to produce behavior from the probabilistic update equations (5) and (A.1.2), a procedure to convert a probability into action is needed.

At each time step, we calculate the belief state as the marginal of the distribution updated via (A.1) on the most recent cue x_t

$$\begin{aligned}\hat{s} &= P(s_t = 1|\boldsymbol{x}_{\leq t}) \\ &= \sum_{\theta_t} P(s_t = 1, \theta_t|\boldsymbol{x}_{\leq t}) \\ &= \sum_{\theta_t} P(s_t = 1|\theta_t, \boldsymbol{x}_{\leq t}) P(\theta_t|\boldsymbol{x}_{\leq t}),\end{aligned}$$

where we evolve the agent under the full or naive update prescribed by the Bayesian equations above.

We then choose to act as soon as the belief suggests maximum expected reward rate under the current context distribution $P(\theta_t|\boldsymbol{x}_{\leq t}) = \sum_{\theta_t} P(s_t, \theta_t|\boldsymbol{x}_{\leq t})$ of the agent. Explicitly, we calculate this threshold by first getting the optimal action time $\tau^\star$ under the current belief as

$$\tau^\star = \text{argmax}_\tau \, \mathbb{E}_{\theta_t} \left[r(\tau; \theta_t) \right]$$

via r from (A.1). Then, we get the average probability to be safe after this waiting time $\tau^\star$ from (A.1) to get

$$\hat{s}_t^\star = \mathbb{E}_{\theta_t} \left[P(s_t = 1|\boldsymbol{x}_{t-\tau:t} = 1, \, \theta_t) \right].$$

Note that while this policy is technically is deterministic for any given *complete* trial history $\boldsymbol{x}_{\leq t}$, the variance in waiting time τ visible in Figure 2 resulted from the within-trial fluctuations in the cues that precede the string of consecutive goes $\boldsymbol{x}_{t-\tau:t} = 1$.

The state and context estimators of the Bayesian agent were computed as averages with respect to the updated joint distribution from (5).

Handling post-action task structure. Whenever the Bayesian agent acted, we reflect this intervention to the world in the probabilistic estimate, which is inherent to the task design in Appendix A.3. Upon action, we transition the agent to a new auxiliary state $s = 2$ that reflects the ITI. This state had a variable duration in the mouse task, which we account for by a simple model of leaving probability

$$P(s|s' = 2, \tau_{a:}) = \begin{cases} 0, & \text{if } s = 1 \\ f(\tau_{a:}), & \text{if } s = 0 \\ 1 - f(\tau_{a:}), & \text{if } s = 2 \end{cases},$$

where $\tau_{a:}$ denotes the time elapsed after action and

$$f(\tau_{a:}) = \begin{cases} 0, & \text{if } \tau_{a:} < 5 \\ 1, & \text{if } \tau_{a:} \geq 25 \\ \frac{1}{25 - \tau_{a:}}, & \text{otherwise.} \end{cases}$$

The function f is in direct correspondence to the mouse task, where the ITI had uniform length between 5 and 25 with uniform distribution.

Differing train- and test-time statistics. Finally, because the network was trained on a world of a different timescale then it was tested on, we allow for parameters that reflect the inference of the network of the changed timescale ϵ of the world. We modelled this by choosing the timescale of the context transition matrix to be $\boldsymbol{T}_{\theta \neq \theta'} = \epsilon = 0.004$, but keep all other aspects of the world fixed. Note that this change by choice does *not* reflect the true statistics of the world, which are inaccessible to the agent because they weren't seen during training. To visualize updates that do not lead to a complete reset after a *nogo*, we augment the original task by a 10^{-3} probability to observe a *nogo* even in the safe state. This allows for finite updates in Figure 4.

A.3.2 Data collection and analysis

In total, we collected three sets of data were collected to produce the results in the main text, contained in three Jupyter notebooks that produce the figures.

For Figure 3, we generated a specific θ trajectory with each θ lasting, on average, 200 steps. To average over many repetitions of a specific θ trajectory, we fixed the realization of the θ_t trajectory and averaged many realizations of cue trajectories x_t. We generated this function by creating sampling the durations d_θ of each context according to an exponential distribution $d_\theta \sim \exp(\epsilon)$, with a rate parameter $\epsilon = \frac{\lambda}{100}$ much slower than the timescale of s_t. For each of these durations, we sampled a single θ from a range of 50 equidistant values between 0.1 and 0.9 (without repetition). The resulting trajectory was repeated until 200,000 trials of behavior were completed.

For Figure 2, we generated 5000 blocks of 50 trials. At the onset of each block θ was sampled from a list of 6 equidistant values between 0.1 and 0.9 (allowing for repetition).

For Figures 4a and 4b we generated 5000 blocks of 50 trials. At the onset of each block θ was sampled from a list of 100 equidistant values between 0.1 and 0.9 (allowing for repetition). With 2% probability, we planted a special trial of $x_{0:5} = (0, 1, 1, 1, 1, 0)$ into an *unsafe* period in order to systematically visualize and average the effect of a *nogo* after consecutive *go*s. Figures 4a and 4b were produced by visualizing this planted-trial data prior to action time to avoid interference with data that originates from the post-action ITI.

All analysis presented in the main text was performed on the same trained network parameters. The behavioral data collected from the networks included whether reward was obtained in a trial, the first action time relative to trial onset, and the waiting time. This formed the basis for further analysis. Trial waiting time was calculated as the time elapsed from the last *nogo* before the first action. If no *nogo*s occurred prior to action, waiting time was calculated as the time from trial onset. In order to validate this calculation all trials in the testing phase, all trials ended with a *nogo*. Empirical reward rates were computed as the average reward per trial divided by the average action time per trial plus the expected ITI duration.

The stimulus-specific vector fields of Figure 4c were computed using `numpy.diff` with `prepend=0`, and the preceding step data was used as the starting location. As a result, the vector fields shown reveal only the effect of the stimulus in the time step in which they were a direct input. Trial data after action was removed from this analysis to avoid the ITI state from confounding. To elucidate the response at each location of the state space, we binned and averaged the data at each presented pair of $(\hat{s}, \hat{\theta})$.

State estimates and behavior comparisons had different requirements and were performed on different trials. For comparing the latent variable estimates, we showed the Bayesian agent the exact inputs seen by the network. This allowed for the precise time-step averaging in Figure 4. In this analysis the Bayesian agent experienced action when the network acted enabled us to filter out experience prior to action.

For comparing behavioral policies, identical inputs were impossible due to the effect of behavior on the state. As a result the Bayesian agent experienced new trials generated from the same θ statistics as the network. This allowed for the trial average behavioral analyses in Figures 3 and2.

For both network and Bayesian agent, the state estimate $\hat{s_t}$ was computed via $-\log(1-P(s_t|\boldsymbol{x} \leq t)$ for numerical stability. For the Bayesian agent, $P(s_t|\boldsymbol{x}_{\leq t})$ was clipped to the interval $[10^{-8}, 1 - 10^{-8}]$.

A.3.3 REGRESSION

We asked whether the estimates of state and context decoded from network activity would resemble the dynamics of the Bayesian estimates. To address this question it was important not to regress directly on the Bayesian estimates. Instead regression was performed on the ground truth state and context.

Regression was performed on all activity in the Actor LSTM, consisting of an Input Gate, Output Gate, Cell Gate, Forget Gate, Short Term Memory Stream and Long Term Memory Stream. Cross-validation was performed by reserving half of the testing episodes for training the decoders, while analysis was performed on the other half of the data that was not regressed on. Additionally, catch trials in which specific stimuli were planted were not seen by the decoder. To train the state decoder we used scikit-learn's LogisticRegression and to train the context decoder we used scikit-learn's LinearRegression. No hyper-parameter tuning was performed on the regressors.

Correlations presented in Figures 2 and 5 were Pearson correlations.

A.4 ANIMAL TRAINING

11 water-deprived C57BL/6 male head-fixed mice were trained in a novel auditory change detection task. Licking in the *unsafe* state was considered premature, leading to an omission of the reward and a 7 second 'time-out' penalty. In each session there were 2-3 different θ values introduced in blocks of 30-50 trials. Sessions were terminated when the mice were satiated and stopped licking. We found that the behavior of the mice was consistent throughout the session. We consider for analysis only the first 300 trials in each session. Performance was monitored, and delivery of sounds and water reinforcement were controlled using computer data acquisition hardware (National Instruments) and custom software written in Matlab (Mathworks). The mice licking was tracked on-line using Bonsai, a visual programming framework.

All experimental procedures were conducted in accordance with our university's animal care and use committee.