Gated recurrence enables simple and accurate sequence prediction in stochastic, changing, and structured environments

Cedric Foucault[1,2,*] and Florent Meyniel[1,*]

1. Cognitive Neuroimaging Unit, INSERM, CEA, Université Paris-Saclay, NeuroSpin center, 91191 Gif/Yvette, France.
2. Sorbonne Université, Collège Doctoral, F-75005 Paris, France.
* Corresponding authors: cedric.foucault@gmail.com and florent.meyniel@cea.fr

Abstract

From decision making to perception to language, predicting what is coming next is crucial. It is also challenging in stochastic, changing, and structured environments; yet the brain makes accurate predictions in many situations. What computational architecture could enable this feat? Bayesian inference makes optimal predictions but is prohibitively difficult to compute. Here, we show that a specific recurrent neural network architecture enables simple and accurate solutions in several environments. To this end, a set of three mechanisms suffices: gating, lateral connections, and recurrent weight tuning. Like the human brain, such networks develop internal representations of their changing environment (including estimates of the environment's latent variables and the precision of these estimates), leverage multiple levels of latent structure, and adapt their effective learning rate to changes without changing their connection weights. Being ubiquitous in the brain, gated recurrence could therefore serve as a generic building block to predict in real-life environments.

[150 words]

Introduction

Being able to correctly predict what is coming next is advantageous: it enables better decisions[1,2], a more accurate perception of our world, and faster reactions[3–7]. Prediction is informed by a sequence of past observations and formally corresponds to a statistical inference, which aims to estimate latent features of the environment (e.g. the probability of a stimulus). This inference is difficult in real life because environments pose specific challenges: they are often partly random, changing, and structured in different ways. Yet, the brain makes reasonable predictions in many situations[2,3,5,8]. What computational mechanisms could enable the brain to overcome these challenges to prediction?

We start by unpacking two specific challenges. First, the joint presence of randomness and changes (i.e., the non-stationarity of the stochastic process generating the observations) poses a well-known tension between stability and flexibility[9–11]. Randomness in observations requires integrating information over time to derive a stable estimate. However, when a change in the estimated variable is suspected, it is better to limit the integration of past observations to update the estimate more quickly. The prediction should thus be adaptive, i.e. dynamically adjusted to promote flexibility in the face of changes and stability otherwise. Past studies have shown that the brain does so in many contexts: perception[12,13], homeostatic regulation[14,15], sensorimotor control[16,17], and reinforcement learning[1,9,10,18].

Second, the structure of our environment can involve complex relationships. For instance, the sentence beginnings "what science can do for you is ..." and "what you can do for science is ..." call for different endings even though they contain the same words, illustrating that prediction takes into account the ordering of observations. Such structures appear not only in human language but also animal communication[5,19–21], and all kinds of stimulus-stimulus and stimulus-action associations in the world[1,4,10,22]. Such a structure is latent (i.e. not directly observable) and it governs the relationship between observations (e.g. words forming a sentence, stimulus-action associations). These relationships must be leveraged by the prediction, making it more difficult to compute. In sum, the randomness, changes, and latent structure of real-life environments pose two major challenges: that of adapting to changes and that of leveraging the latent structure.

Two commonly used approaches offer different solutions to these challenges. The Bayesian approach allows to derive statistically optimal predictions for a given environment knowing its underlying generative model. This optimal solution is a useful benchmark and has some descriptive validity since, in some contexts, organisms behave close to optimally[23,24] or exhibit several qualitative aspects of the optimal solution[9,25,26]. However, a specific Bayes-optimal solution only applies to a specific generative model (or class of models[27]). This mathematical solution also does not in general lead to an algorithm of reasonable complexity[28,29]. Bayesian inference therefore says little about the algorithms that the brain could use, and the biological basis of those computations remain mostly unknown with only a few proposals highly debated[30–32].

Opposite to the Bayes-optimal approach is the heuristics approach: solutions that are easy to compute and accurate in specific environments[33]. However, heuristics lack generality: their performance can be quite poor outside the environment that suits them. In addition, although simple, their biological implementation often remains unknown (besides the delta-rule[34–36]).

Those two approaches leave open the following questions: Is there a general, biologically feasible architecture that enables, in different environments, solutions that are simple, effective, and that reproduce the qualitative aspects of prediction observed in organisms? If so, what are its essential mechanistic elements?

Our approach stands in contrast with the elegant closed-form but intractable mathematical solutions offered by Bayesian inference, and the simple but specialized algorithms offered by heuristics. Instead, we focus on general mechanisms under the constraints of feasibility and simplicity. We used recurrent neural networks because they can offer a generic, biologically feasible architecture able to realize different prediction algorithms (see Ref.[37,38] and Discussion). We tested their generality using different environments. To determine whether simple and effective solutions are possible and provide mechanistic insight, we considered different architectures that varied in size and mechanisms, and trained networks to approach their best possible prediction algorithm in each environment. We treated the training procedure as a methodological step without claiming it to be biologically plausible. Only the sequences of observations were fed into the training. To provide interpretability, we inspected the networks' internal model and representations, and tested specific optimal aspects of their behavior—previously reported in humans[25,26,39,40]—which demonstrate the ability to adapt to changes and leverage the latent structure of the environment.

Results

The framework: sequence prediction and network architectures

All our analyses confront simulated agents with the same general problem: sequence prediction. It consists in predicting, at each time step in a sequence, the probability distribution over the value of the next observation given the previous observations (here we used binary observations coded as '0' and '1') (Fig. 1a). The environment generates the sequence, and the agent's goal is to make the most accurate predictions possible in this environment. Below, we introduce three environments. All of them are stochastic (observations are governed by latent probabilities) and changing (these latent probabilities change across time), and thus require dynamically adapting the stability-flexibility tradeoff. They also feature increasing levels of latent structure that must be leveraged, making the computation of predictions more complex.

How do agents learn to make predictions that fit a particular environment? In real life, agents often do not benefit from any external supervision and must rely only on the observations. To do so, they can take advantage of an intrinsic error signal that measures the discrepancy between their prediction and the actual value observed at the next time step. We adopted this learning paradigm (often called unsupervised, self-supervised, or predictive learning in machine learning[41,42]) to train our agents in silico. We trained the agents by exposing them to sequences generated by a given environment and letting them tune their parameters to improve their prediction (see Methods).

During testing, we kept the parameters of the trained agents frozen, exposed them to new sequences, and performed targeted analyses to probe whether they exhibit specific capabilities and better understand how they solve the problem.

Our investigation focuses on a particular class of agent architectures known as recurrent neural networks. These are well suited for sequence prediction because recurrence allows to process inputs sequentially while carrying information over time in recurrent activity. The network architectures we used all followed the same three-layer template, consisting of one input unit whose activity codes for the current observation, one output unit whose activity codes for the prediction about the next observation, and a number of recurrent units that are fed by the input unit and project to the output unit (Fig. 1b). All architectures had self-recurrent connections.

We identified three mechanisms of recurrent neural network architectures that endow a network with specific computational properties which have proven advantageous in our environments (Fig. 1c). One

mechanism is gating, which allows multiplicative interactions between the activities of units. A second mechanism is lateral connectivity, which allows the activities of different recurrent units to interact with each other. A third mechanism is the tuning of recurrent connection weights during training, which allows the dynamics of recurrent activities to be adjusted to the training environment.

To get mechanistic insight, we compared the architecture that included all three mechanisms, here called 'gated recurrent' but also known as GRU[43,44], to alternative architectures that were deprived of one of the three mechanisms but maintained the two others (Fig. 1d; see Methods for equations). When deprived of gating, multiplicative interactions between activities are removed, and the architecture reduces to that of a simple recurrent neural network also known as the Elman network[45]. When deprived of lateral connections, the recurrent units become independent of each other, thus each recurrent unit acts as a temporal filter on the input observations (with possibly time-varying filter weights thanks to gating). When deprived of recurrent weight tuning, the dynamics of the recurrent activities become unattuned to the environment, since the only parameters that can be tuned during training are those of the output unit; this architecture is thus one form of reservoir computing[46]. In the results below, unless otherwise stated, the networks all had 11 recurrent units (the smallest network size beyond which the gated recurrent network showed no substantial increase in performance in any of the environments), but the results across architectures are robust to this choice of network size (see the last section of the Results).

Performance in the face of changes in latent probabilities

We designed a first environment to investigate the ability to handle changes in a latent probability (Fig. 2a; see Supplementary Fig. 1 for a graphical model). In this environment we used the simplest kind of latent probability: p1, the probability of occurrence (or base rate) of the observation being 1 (note that p0=1-p1), here called 'unigram probability'. The unigram probability suddenly changed from one value to another at so-called 'change points', which could occur at any time, randomly with a given fixed probability.

This environment, here called 'changing unigram environment', corresponds for instance to a simple oddball task[47–49], or the probabilistic delivery of a reward with abrupt changes in reward probabilities[9,50]. In such an environment, predicting accurately is difficult due to the stability-flexibility tradeoff induced by the stochastic nature of the observations (governed by the unigram probability) and the possibility of a change point at any moment.

To assess the networks' prediction accuracy, we compared the networks with the optimal agent for this specific environment (i.e. the optimal solution to the prediction problem determined using Bayesian inference; see Methods). We also compared them to two types of heuristics which perform very well in this environment: the classic 'delta-rule' heuristic[1,34] and the more accurate 'leaky' heuristic[26,51–53] (see Methods for details). To test the statistical reliability of our conclusions, we trained separately 20 agents of each type (each type of network and each type of heuristic).

We found that even with as few as 11 units, the gated recurrent networks performed quasi-optimally. Their prediction performance was 99% of optimal (CI ±0.1%), 0% corresponding to chance level (Fig. 2b). Being only 1% short of optimal, the gated recurrent networks outperformed the delta rule and leaky agents, which performed 10 times and 5 times further from optimal, respectively (Fig. 2b).

For mechanistic insight, we tested the alternative architectures deprived of one mechanism. Without either gating, lateral connections, or recurrent weight tuning, the average performance was respectively 6 times, 4 times, and 12 times further from optimal (Fig. 2b), i.e. the level of a leaky agent or worse. The drops in performance remain similar when considering only the best network of each architecture instead of the average performance (Fig. 2b, compare rightmost dots across rows).

These results show that small gated recurrent networks can achieve quasi-optimal predictions and that the removal of one of the mechanisms of the gated recurrent architecture results in a systematic drop in performance.

Adaptation to changes through the adjustment of the effective learning rate

In a changing environment, the ability to adapt to changes is key. Networks exposed to more changing environments during training updated their predictions more overall during testing, similarly to the optimal agent (see Supplementary Fig. 2) and, to some extent, humans[9,54]. At a finer timescale, the moment-by-moment updating of the predictions also showed sensible dynamics around change points.

Fig. 3a illustrates a key difference in behavior between, on the one hand, the optimal agent and the gated recurrent network, and on the other hand, the heuristic agents: the dynamics of their update differ. This difference is particularly noticeable when recent observations suggest that a change point has just occurred: the optimal agent quickly updates the prediction by giving more weight to the new observations; the gated recurrent network behaves the same but not the heuristic agents. We formally tested this dynamic updating around change points by measuring the moment-by-moment effective learning rate, which normalizes the amount of update in the prediction by the prediction error (i.e. the difference between the previous prediction and the actual observation; see Methods, equation (2)).

Gated recurrent networks turned out to adjust their moment-by-moment effective learning rate as the optimal agent did, showing the same characteristic peaks, at the same time and with almost the same amplitude (Fig. 3b, top plot). By contrast, the effective learning rate of the delta-rule agents was (by construction) constant, and that of the leaky agents changed only marginally.

When one of the mechanisms of the gated recurrence was taken out, the networks' ability to adjust their effective learning rate was greatly degraded (but not entirely removed) (Fig. 3b, bottom plots). Without gating, without lateral connections, or without recurrent weight tuning, the amplitude was lower (showing both a lower peak value and a higher baseline value), and the peak occurred earlier.

This shows that gated recurrent networks can reproduce a key aspect of optimal behavior: the ability to adapt the update of their prediction to change points, which is lacking in heuristic agents and alternative networks.

Reliance on precision-weighting for adaptation to changes

Beyond behavior, we sought to determine whether a network's ability to adapt to changes relied on idiosyncratic computations or followed the more general principle of precision-weighting derived from probability theory. According to this principle, the precision of the current prediction (as per equation (3) in Methods) should influence the weight of the current prediction relative to the next observation in the updating process: for a given prediction error, the lower the precision, the higher the subsequent effective learning rate. This precision-weighting principle results in an automatic adjustment of the effective learning rate in response to a change, because the precision of the prediction decreases when a change is suspected.

In line with this principle, human participants can estimate not only the prediction but also its precision as estimated by the optimal agent[25,55], and this precision indeed relates to the participants' effective learning rate[39,40,56].

We tested whether a network could represent this optimal precision too, by trying to linearly read it from the network's recurrent activity (Fig. 4a). Note that the networks were trained only to maximize prediction accuracy (not to estimate precision). Yet, in gated recurrent networks, we found that the read precision on

left-out data was highly accurate (Fig. 4b, left plot: the median Pearson correlation with the optimal precision is 0.82), and correlated with their subsequent effective learning rate as in the optimal agent (Fig. 4b, right plot: the median correlation for gated recurrent networks is -0.79; for comparison, it is -0.88 for the optimal agent).

We then tested whether this internal representation of precision causally regulated the effective learning rate in the networks, rather than simply being correlated with it, using a perturbation experiment. We designed perturbations of the recurrent activity that induced a controlled change in the read precision while leaving the networks' current prediction unchanged. These perturbations caused significant changes in the networks' subsequent effective learning rate, commensurate with the induced change in precision, as predicted by the principle of precision-weighting (Fig. 4c, middle plot). Importantly, this causal relationship was abolished in the alternative networks that lacked one of the mechanisms of the gated recurrent architecture (Fig. 4c, right three plots; the slope of the effect was significantly different between the gated recurrent network group and any of the alternative network groups, two-tailed two independent samples t-test, all $t(38) > 4.1$, all $p < 0.001$, all Cohen's $d > 1.3$).

These results show that the gated recurrent networks' ability to adapt to changes indeed relies on their precision-dependent updating and that all three mechanisms of the gated recurrence are necessary for such precision-weighting to arise.

Leveraging and internalizing a latent structure: bigram probabilities

While the changing unigram environment already covers many tasks in the behavioral and neuroscience literature, real-world sequences often exhibit more structure. To study the ability to leverage such structure, we designed a new stochastic and changing environment in which the sequence of observations is no longer generated according to a single unigram probability, $p(1)$, but two 'bigram probabilities' (also known as transition probabilities), $p(0|0)$ and $p(1|1)$, which denote the probability of occurrence of a 0 after a 0 and of a 1 after a 1, respectively (Fig. 5a; see Supplementary Fig. 1 for a graphical model). These bigram probabilities are also changing randomly, with independent change points.

This 'changing bigram environment' is well motivated because there is ample evidence that bigram probabilities play a key role in sequence knowledge in humans and other animals[5] even in the face of changes[25,57].

We assessed how well the networks could leverage the latent bigram structure after having been trained in this environment. For comparison, we tested the optimal agent for this environment as well as two groups of heuristics: delta-rule and leaky estimation of unigram probabilities (as in Fig. 2b), and now also delta rule and leaky estimation of bigram probabilities (see Methods for details).

The gated recurrent networks achieved 98% of optimal prediction performance (CI ±0.3%), outperforming the heuristic agents estimating bigram probabilities, and even more so those estimating a unigram probability (Fig. 5c). To demonstrate that this was due to their internalization of the latent structure, we also tested the gated recurrent networks that had been trained in the changing unigram environment: their performance was much worse (Supplementary Fig. 3).

At the mechanistic level, this ability to leverage the latent bigram structure requires all three mechanisms of the gated recurrence. Not only does the performance drop when one of these mechanisms is removed (Fig. 5c), but also this drop in performance is much larger than that observed in the changing unigram environment (without gating: -11.1% [CI ±1.2% calculated by Welch's t-interval] in the bigram environment vs. -5.5% [CI ±0.6%] in the unigram environment, without lateral connections: -20.5% [CI ±3.7%] vs. -2.9% [CI ±0.2%]; without recurrent weight tuning: -30.4% [CI ±1.7%] vs. -11.0% [CI ±2.1%]; for

every mechanism, there was a significant interaction effect between the removal of the mechanism and the environment on performance, all $F(1,76) > 69.8$, all $p < 0.001$).

Fig. 5b illustrates the gated recurrent networks' ability to correctly incorporate the bigram context into its predictions compared to networks lacking one of the mechanisms of the gated recurrence. While a gated recurrent network aptly changes its prediction from one observation to the next according to the preceding observation as the optimal agent does, the other networks fail to show such context-dependent behavior, sometimes even changing their prediction away from the optimal agent.

Altogether these results show that recurrent networks can leverage the latent bigram structure, but only when equipped with all three mechanisms of the gated recurrence.

Is the networks' representation of the latent bigram structure impenetrable or easily accessible? We tested the latter possibility by trying to linearly read out the optimal estimate of each of the latent bigram probabilities from the recurrent activity of a gated recurrent network (see Methods). Arguing in favor of an explicit representation, we found that the read estimates of each of the latent bigram probabilities on left-out data were highly accurate (Pearson correlation with the optimal estimates, median and CI: 0.97 [0.97, 0.98] for each of the two bigram probabilities).

In addition to the point estimates of the latent bigram probabilities, we also tested whether a network maintained some information about the precision of each estimate. Again, we assessed the possibility to linearly read out the optimal precision of each estimate and found that the read precisions on left-out data were quite accurate (Pearson correlation with the optimal precisions, median and CI: 0.77 [0.74, 0.78] for one bigram probability and 0.76 [0.74, 0.78] for the other probability).

Fig. 5d illustrates the striking resemblance between the estimates read from a gated recurrent network and the optimal estimates. Furthermore, it shows that the network successfully disentangles one bigram probability from the other since the read estimates can evolve independently from each other (for instance during the first 20 time steps, the value for 1|1 changes while the value for 0|0 does not, since only 1s are observed). It is particularly interesting that both bigram probabilities are simultaneously represented, given that only one of them is relevant for the moment-by-moment prediction read by the network's output unit (whose weights cannot change during the sequence).

We conclude that gated recurrent networks internalize the latent bigram structure in such a way that both bigram probabilities are available simultaneously, even though only one of the two is needed at any one time for the prediction.

Leveraging a higher-level structure: inference about latent changes

In real life, latent structures can also exhibit different levels that are organized hierarchically[25,58,59]. To study the ability to leverage such a hierarchical structure, we designed a third environment in which, in addition to bigram probabilities, we introduced a higher-level factor: the change points of the two bigram probabilities are now coupled, rather than independent as they were in the previous environment (Fig. 6a; Supplementary Fig. 1 shows the hierarchical structure). Due to this coupling, from the agent's point of view, the likelihood that a change point has occurred depends on the observations about both bigrams. Thus, optimal prediction requires the ability to make a higher-level inference: having observed that the frequency of one of the bigrams has changed, one should not only suspect that the latent probability of this bigram has changed but also transfer this suspicion of a change to the latent probability of the other bigram, even without any observations about that bigram.

Such a transfer has been reported in humans[26]. A typical situation is when a streak of repetitions is encountered (Fig. 6b): if a long streak of 1s was deemed unlikely, it should trigger the suspicion of a

change point such that p(1|1) is now high, and this suspicion should be transferred to p(0|0) by partially resetting it. This reset is reflected in the change between the prediction following the 0 just before the streak and that following the 0 just after the streak (Fig. 6b, $|p_{after} - p_{before}|$).

We tested the networks' ability for higher-level inference in the same way, by exposing them to such streaks of repetitions and measuring their change in prediction about the unobserved bigram before and after the streak. More accurately, we compared the change in prediction of the networks trained in the environment with coupled change points to that of the networks trained in the environment with independent change points, since the higher-level inference should only be made in the coupled case.

We found that gated recurrent networks trained in the coupled environment changed their prediction about the unobserved bigram significantly more than networks trained in the independent environment, and this was true across a large range of streak lengths (Fig. 6c, top plot). The mere presence of this effect is particularly impressive given that the coupling makes very little difference in terms of raw performance (Supplementary Fig. 3, the networks trained in either the coupled or the independent environment perform very similarly when tested in either environment). It seems that all mechanisms of the gated recurrence are needed to achieve this higher-level inference since the networks deprived of either gating, lateral connections, or recurrent weight tuning did not show any effect, no matter the streak length (Fig. 6c, bottom three plots; for every mechanism, there was a significant interaction effect between the removal of the mechanism and the training environment on the change in prediction over networks and streak lengths, all $F(1,6076) > 57.8$, all $p < 0.001$).

These results show that only gated recurrent networks can leverage the higher level of structure where the change points of the latent probabilities are coupled.

Simple solutions are uniquely enabled by the combination of gating, lateral connections, and recurrent weight tuning

Finally, we highlight the small number of units required to perform quasi-optimally in the increasingly structured environments that we tested: the above-mentioned results were obtained with 11 recurrent units. It turns out that gated recurrent networks can reach a similar performance with even fewer units, especially in simpler environments (Fig. 7a and 7b, left plot). For instance, in the unigram environment, gated recurrent networks reach 99% of their asymptotic performance with no more than 3 units.

By contrast, without either gating, lateral connections, or recurrent weight tuning, the networks are unable to reach such performance, even if they are provided with more units to match the number of tuned parameters in the 11-unit gated recurrent networks (Fig. 7a and 7b, right three plots, the twin x-axes indicate the number of units and tuned parameters). Note that, in terms of computational complexity, the number of units is a fair measure of space complexity regardless of the architecture, since it always equals to the dimensionality of the activity and the memory capacity of the network (note that there is no memory cell in the gated units used here, unlike LSTM units). What varies across architectures is the number of tuned parameters, i.e. the degrees of freedom that can be used during training to achieve different dynamics. Still, the conclusion remains the same when an alternative network exceeds the complexity of an 11-unit gated recurrent network in both its number of units and its number of tuned parameters.

Therefore, it is not the mere additional number of tuned parameters, but the specific computational properties provided by the combination of the three mechanisms that allow effective solutions to be realized with low space complexity.

Discussion

We have shown that the gated recurrent architecture enables simple and effective solutions: with only 11 units, the networks perform quasi-optimally in environments fraught with randomness, changes, and different levels of latent structure. Moreover, these solutions reproduce several aspects of optimality observed in organisms, including the adaptation of their effective learning rate, the ability to represent the precision of their estimation and to use it to weight their updates, and the ability to represent and leverage the latent structure of the environment. By depriving the architecture of one of its mechanisms, we have shown that three of them are necessary to achieve such solutions: gating, lateral connections, and the tuning of recurrent weights to the environment.

Can small neural networks behave like Bayesian agents?

A central and much-debated question in the scientific community is whether the brain can perform Bayesian inference[60–69,10,54]. From a computational viewpoint, there exists no tractable solution (even approximate) for Bayesian inference in an arbitrary environment, since it is NP-hard[28,29]. Being a bounded agent[70,71], the brain cannot solve Bayesian inference in its most general form. The interesting question is whether the brain can perform Bayesian inference in some environments that occur in real life. More precisely, by “perform Bayesian inference” one usually means that it performs computations that satisfy certain desirable properties of Bayesian inference, such as taking into account a certain type of uncertainty and a certain type of latent structure[23,24,60,62,72–74]. In this study, we selected specific properties and showed that they can indeed be satisfied when using specific (not all) neural architectures.

In the changing unigram and changing bigram environments, our results provide an existence proof: there exist plausible solutions that are almost indistinguishable from Bayesian inference (i.e. the optimal solution). They exhibit qualitative properties of Bayesian inference that have been demonstrated in humans but are lacking in heuristic solutions, such as the dynamic adjustment of the effective learning rate[9,39,40], the internal representation of latent variables and the precision of their estimates[25,55], the precision-weighting of updates[39,40,56], and the ability for higher-level inference[26,58,59].

The performance we obtained with the gated recurrent architecture is consistent with the numerous other successes it produced in other cognitive neuroscience tasks[75–77]. Our detailed study reveals that it offers quasi-optimal low-complexity solutions to new and difficult challenges, including those posed by bigram and higher-level structures and latent probabilities that change unpredictably anywhere in the unit interval. We acknowledge that further generalization to additional challenges remains to be investigated, including the use of more than two categories of observations or continuous observations, and latent structures with longer range dependencies (beyond bigram probabilities).

Minimal set of mechanisms

What are the essential mechanistic elements that enable such solutions? We show that it suffices to have recurrent units of computation equipped with three mechanisms: 1) input, self, and lateral connections which enable each unit to sum up the input with their own and other units’ prior value before a non-linear transformation is applied; 2) gating, which enables multiplicative interactions between activities at the summation step; 3) the tuning of connections during training.

One of the advantages of such mechanisms is their generic character: they do not include any components specifically designed to perform certain probabilistic operations or estimate certain types of latent variables, as often done in neuroscience[30,78–82]. In addition, they allow adaptive behavior only through recurrent activity dynamics, without involving synaptic plasticity as in other models[35,83–85]. This distinction has implications for the timescale of adaptation: in the brain, recurrent dynamics and synaptic

plasticity often involve short and long timescales, respectively. Our study supports this view: recurrent dynamics allow the networks to quickly adapt to a given change in the environment (Fig. 3), while synaptic plasticity allows the training process to tune the speed of this adaptation to the frequency of change of the environment (Supplementary Fig. 2).

Our findings suggest that these mechanisms are not only sufficient but necessary (within the set of biologically feasible mechanisms we tested; other mechanisms exist): without one, a reasonably sized network is unable to perform close to optimally or exhibit the desired properties in the studied environments. This demonstrates the importance of considering resource constraints to reveal the benefits of an architecture (although a standard network architecture may have universal approximation properties, the size needed to achieve them can be exponential in the dimension of the state space[86–88]; in our case, the state space underlying our prediction problem is infinite-dimensional since one state includes a continuous probability density function).

Biological implementations of the mechanisms

What biological elements could implement the mechanisms of the gated recurrence? Recurrent connections are ubiquitous in the brain[89,90]; the lesser-known aspect is that of gating.

In neuroscience, gating is often encountered in studies of attention, and gating mechanisms are often explained in terms of inhibition/disinhibition, neural oscillations, or neuromodulation. In inhibition/disinhibition, gating is realized by inhibitory (GABAergic) neurons which let more or less signal through in a given pathway. This pathway can be microscopic (e.g. a dendrite of a pyramidal neuron)[91,92], or macroscopic (e.g. a thalamocortical pathway gated by the basal ganglia)[93–95]. . In neural oscillations, experiments have shown that activity in certain frequency bands (typically, alpha and beta) can gate behavioral and neuronal responses to the same stimulus[96–101]. One of the most influential accounts is known as “pulsed inhibition”[102–104]: a low-frequency signal periodically inhibits a high-frequency signal, effectively silencing the high-frequency signal when the low-frequency signal exceeds a certain threshold. Finally, in neuromodulation, the binding of certain neuromodulators to the certain receptors of a synapse changes the gain of its input-output transfer function, thus changing its effective weight. This has been demonstrated in neurophysiological studies implicating noradrenaline[105–107], dopamine[106,108–110], and acetylcholine[111,112] (see review in Ref.[113]).

We claim that gated recurrence provides plausible solutions for the brain because its mechanisms can all be biologically implemented and lead to inexpensive solutions. However, given their multiple biological realizability, the mapping between artificial units and biological neurons may not be straightforward: one unit may map to a large population of neurons (e.g. a brain area), or even to a microscopic, subneuronal component (e.g. the dendritic level).

Weight tuning: its role and possible biological counterpart

Regarding the weight tuning, our results highlight that it is important for recurrent weights to be tuned to the environment (unlike reservoir computing[46]), but we make no claims about the biological process that lead to such a weight tuning in brains. This tuning could occur during development[6], the life span[114], or the evolution process[115] (these possibilities are not mutually exclusive). Although our training procedure may not be accurate for biology as a whole, two aspects of it may be informative for future research. First, it relies only on the observation sequence (no supervision or reinforcement), leveraging prediction error signals, which have been found in the brain in many studies[36,116,117]. Importantly, in predictive coding[65], the computation of prediction errors is part of the prediction process; here we are suggesting that it may also be part of the training process (as argued in Ref.[118]). Second, relatively few iterations of training suffice

(Supplementary Fig. 4, in the order of 10–100; for comparison, Ref.[77] reported training for 40,000 episodes in an environment similar to ours).

Implications for experimentalists

If already tuned gated recurrent networks exist in the brain, then one can be used in a new but similar enough environment without further tuning. This is an interesting possibility because, in laboratory experiments mirroring our study, humans perform reasonably well with almost no training but explicit task instructions given in natural language, along with a baggage of prior experience[8,25,26,119,120]. In favor of the possibility to reuse an existing solution, we found that a gated recurrent network can still perform well in conditions different from those it was trained in: across probabilities of change points (Supplementary Fig. 2) and latent structures (Supplementary Fig. 3, from bigram to unigram).

One of our findings may be particularly interesting to experimentalists: in a gated recurrent network, the representations of latent probabilities and the precision of these probability estimates (sometimes referred to as confidence[25,55], estimation uncertainty[56,121], or epistemic uncertainty[15,122,123]) are linearly readable from recurrent activity, the form of decoding most frequently used in neuroscience[124,125]. These representations arise spontaneously, and their emergence seems to come from the computational properties of gated recurrence together with the need to perform well in a stochastic and changing environment. This yields an empirical prediction: if such networks can be found in the brain, then latent probability estimates and their precision should also be decodable in brain signals, as already found in some studies[56,121,126–129].

Conclusion

We conclude that the gated recurrent architecture provides many computational advantages for predicting in stochastic, changing, and structured environments, and thus could serve as a generic building block in the brain.

Methods

Sequence prediction problem

The sequence prediction problem to be solved is the following. At each time step, an agent receives as input a binary-valued 'observation', $x_t \in \{0,1\}$, and gives as output a real-valued 'prediction', $p_t \in [0,1]$, which is an estimate of the probability that the value of the next observation is equal to 1, $p(x_{t+1}=1)$. Coding the prediction in terms of the observation being 1 rather than 0 is inconsequential since one can be deduced from the other: $p(x_{t+1}=1) = 1 - p(x_{t+1}=0)$. The agent's objective is to make predictions that maximize the (log) likelihood of observations in the sequence, which technically corresponds to the negative binary cross-entropy cost function:

$$L(p;x) = \sum_{t=0}^{T-1} \log\left[x_{t+1}p_t + (1-x_{t+1})(1-p_{t+1})\right] \quad \textbf{(1)}$$

Network architectures

All network architectures consist of a binary input unit, which codes for the current observation, one recurrent layer (sometimes called hidden layer) with a number N of recurrent units, and an output unit, which represents the network's prediction. Unless otherwise stated, N=11. At every time step, the recurrent unit i receives as input the value of the observation x_t and the previous activation values of the recurrent units *j* that connect to *i* $h_{j,t-1}$. It produces as output a new activation value $h_{i,t}$, which is a real number. The output unit receives as input the activations of all of the recurrent units and produces as output the prediction p_t.

The parameterized function of the output unit is the same for all network architectures:

$$p_t = \sigma\left(\sum_{i=1}^{N} w_{hp,i} h_{i,t} + b_{hp}\right)$$

where σ is the logistic sigmoid, $w_{hp,i}$ is the weight parameter of the connection from the *i*-th recurrent unit to the output unit, and b_{hp} is the bias parameter of the output unit.

The parameterized function of the recurrent units takes a different form depending on whether gating or lateral connections are included, as described below.

Gated recurrent network. A gated recurrent network includes both gating and lateral connections. The variant of gating used here is GRU[43,44], which introduces, for each recurrent unit *i*, two intermediate variables to its parameterized function: the reset gate r_i and the update gate z_i, both of which have their own set of weights and bias. The update gate controls the extent to which a unit can change its values from one time step to the next, and the reset gate controls the balance between recurrent activity and input activity in case of update; both gates enable multiplicative interactions between the input and recurrent activity as well as between the activities of different recurrent units during the updating process. The activations are calculated as follows:

$$r_{i,t+1} = \sigma\left(w_{xr,i} x_{t+1} + b_{xr} + w_{hr,ii} h_{i,t} + \sum_{j \neq i} w_{hr,ji} h_{j,t} + b_{hr}\right)$$

$$z_{i,t+1} = \sigma\left(w_{xz,i} x_{t+1} + b_{xz} + w_{hz,ii} h_{i,t} + \sum_{j \neq i} w_{hz,ji} h_{j,t} + b_{hz}\right)$$

$$h_{i,t+1} = (1 - z_{i,t+1})\, h_{i,t} + z_{i,t+1} \tanh\left[w_{xh,i} x_{t+1} + b_{xh} + r_{i,t+1}(w_{hh,ii} h_{i,t} + \sum_{j \neq i} w_{hh,ji} h_{j,t}) + b_{hh}\right]$$

$$h_{i,t=-1} = 0$$

where w and b are all weight and bias parameters.

Without gating. Removing the gating mechanism from the gated recurrent network is equivalent to setting the above variables r_i and z_i equal to 1. This simplifies the calculation of the activations to a single equation, which boils down to a weighted sum of the input and the recurrent units' activity before applying a non-linearity, as follows:

$$h_{i,t+1} = \tanh\left[w_{xh,i} x_{t+1} + b_{xh} + w_{hh,ii} h_{i,t} + \sum_{j \neq i} w_{hh,ji} h_{j,t} + b_{hh}\right]$$

Without lateral connections. Removing lateral connections from the gated recurrent network is equivalent to setting the weights $w_{hr,ji}$, $w_{hz,ji}$, and $w_{hh,ji}$ to 0 for all $j \neq i$. This abolishes the possibility of interaction between recurrent units, which simplifies the calculation of the activations as follows:

$$r_{i,t+1} = \sigma\left(w_{xr,i} x_{t+1} + b_{xr} + w_{hr,ii} h_{i,t} + b_{hr}\right)$$
$$z_{i,t+1} = \sigma\left(w_{xz,i} x_{t+1} + b_{xz} + w_{hz,ii} h_{i,t} + b_{hz}\right)$$
$$h_{i,t+1} = (1 - z_{i,t+1})\, h_{i,t} + z_{i,t+1} \tanh\left[w_{xh,i} x_{t+1} + b_{xh} + r_{i,t+1} w_{hh,ii} h_{i,t} + b_{hh}\right]$$

Note that this architecture still contains gating. We could have tested a simpler architecture without lateral connection and without gating; however, our point is to demonstrate that lateral connections are necessary to solve the problem we are interested in with few units, and the result is all the more convincing if the network lacking lateral connections has gating (without gating, it would fail even more dramatically).

Without recurrent weight tuning. The networks referred to as “without recurrent weight tuning” have the same architecture as the gated recurrent networks and differ from them only in the way they are trained. While in the other networks, all of the weights and bias parameters are tuned during the training procedure, for those networks, only the weights and bias of the output unit, w_{hp} and b_{hp}, are tuned during training; other weights and biases are fixed to the value drawn at initialization.

Environments

An environment is characterized by its data generating process, i.e. the stochastic process used to generate a sequence of observations in that environment. Each of the generative processes is described by a graphical model in Supplementary Fig. 1 and further detailed below.

Changing unigram environment. In the changing unigram environment, at each time step, one observation is drawn from a Bernoulli distribution whose probability parameter is the latent variable p_t^{env}. The evolution of this latent variable is described by the following stochastic process.

- Initially, $p_{t=0}^{env}$ is drawn from a uniform distribution on [0,1].
- At the next time step, with probability p_c, p_{t+1}^{env} is drawn anew from a uniform distribution on [0,1] (this event is called a 'change point'), otherwise, p_{t+1}^{env} remains equal to p_t^{env}. The change point probability p_c is fixed in a given environment.

Changing bigram environments. In the changing bigram environments, at each time step, one observation is drawn from a Bernoulli distribution whose probability parameter is either equal to the latent variable $p_{1|1,t}^{env}$, if the previous observation was equal to 1, or to the latent variable $1 - p_{0|0,t}^{env}$ otherwise (at t=0, the previous observation is considered to be equal to 0). The evolution of those latent variables is described by a stochastic process which differs depending on whether the change points are independent or coupled.

- In both cases, initially, $p_{0|0,t=0}^{env}$ and $p_{1|1,t=0}^{env}$ are both drawn independently from a uniform distribution on [0,1].
- In the case of **independent change points**, at the next time step, with probability p_c, $p_{0|0,t+1}^{env}$ is drawn anew from a uniform distribution on [0,1], otherwise, $p_{0|0,t+1}^{env}$ remains equal to $p_{0|0,t}^{env}$. Similarly, $p_{1|1,t+1}^{env}$ is either drawn anew with probability p_c or remains equal to $p_{1|1,t}^{env}$ otherwise, and critically, the occurrence of a change point in $p_{1|1}^{env}$ is independent from the occurrence of a change point in $p_{0|0}^{env}$.
- In the case of **coupled change points**, at the next time step, with probability p_c, $p_{0|0,t+1}^{env}$ and $p_{1|1,t+1}^{env}$ are both drawn anew and independently from a uniform distribution on [0,1], otherwise, both remain equal to $p_{0|0,t}^{env}$ and $p_{1|1,t}^{env}$ respectively.

The changing bigram environment with independent change points and that with coupled change points constitute two distinct environments. When the type of change points is not explicitly mentioned, the default case is independent change points. For conciseness, we sometimes refer to the changing unigram and changing bigram environments simply as “unigram” and “bigram” environments.

In all environments, unless otherwise stated, the length of a sequence is $T = 380$ observations, and the change point probability is $p_c = \frac{1}{75}$, as in previous experiments done with humans participants[25,26].

Optimal solution

For a given environment among the three possibilities defined above, the optimal solution to the prediction problem can be determined as detailed in Ref.[26]. This solution consists in inverting the data-generating process of the environment using Bayesian inference, i.e. computing the posterior probability distribution over the values of the latent variables given the history of observation values, and then marginalizing over that distribution to compute the prediction (which is the probability of the next observation given the history of observations). This can be done using a hidden Markov model formulation of the data-generating process where the hidden state includes the values of the latent variables as well as the previous observation in the bigram case, and using the forward algorithm to compute the posterior distribution over the hidden state. Because it would be impossible to compute the probabilities for the infinitely many possible values of the latent variables in the continuous interval [0,1], we discretized the interval into 20 equal-width bins for each of the latent variables. For a more exhaustive treatment, see Ref.[26] and the online code (https://github.com/florentmeyniel/TransitionProbModel).

Heuristic solutions

The four heuristic solutions used here can be classified into 2×2 groups depending on:

- which kind of variables are estimated: a unigram probability or two bigram probabilities.
- which heuristic rule is used in the calculation of the estimates: the delta-rule or the leaky rule.

The equations used to calculate the estimates are provided below.

Unigram, delta-rule:

$$\widehat{p}_{t+1} = \widehat{p}_t + \alpha\left(x_{t+1} - \widehat{p}_t\right)$$
$$\widehat{p}_{t=-1} = 0.5$$

Unigram, leaky rule:

$$n_{0,t+1} = \alpha n_{0,t} + (1 - x_{t+1})$$
$$n_{1,t+1} = \alpha n_{1,t} + x_{t+1}$$
$$n_{0,t=-1} = n_{1,t=-1} = 0$$
$$\widehat{p}_t = \frac{n_{1,t} + 1}{n_{1,t} + n_{0,t} + 2}$$

Bigrams, delta-rule:

$$\widehat{p}_{0|0,t+1} = \widehat{p}_{0|0,t} + \alpha\left(1 - x_t\right)\left(1 - x_{t+1} - \widehat{p}_{0|0,t}\right)$$
$$\widehat{p}_{1|1,t+1} = \widehat{p}_{1|1,t} + \alpha x_t\left(x_{t+1} - \widehat{p}_{1|1,t}\right)$$
$$\widehat{p}_{0|0,t=-1} = \widehat{p}_{1|1,t=-1} = 0.5$$

Bigrams, leaky rule:

$$
\begin{aligned}
n_{0|0,t+1} &= \alpha n_{0|0,t} + (1 - x_t)(1 - x_{t+1}) \\
n_{01,t+1} &= \alpha n_{01,t} + (1 - x_t)\, x_{t+1} \\
n_{10,t+1} &= \alpha n_{10,t} + x_t (1 - x_{t+1}) \\
n_{1|1,t+1} &= \alpha n_{1|1,t} + x_t x_{t+1} \\
n_{0|0,t=-1} &= n_{01,t=-1} = n_{10,t=-1} = n_{1|1,t=-1} = 0 \\
\widehat{p}_{0|0,t} &= \frac{n_{0|0,t} + 1}{n_{0|0,t} + n_{01,t} + 2} \\
\widehat{p}_{1|1,t} &= \frac{n_{1|1,t} + 1}{n_{1|1,t} + n_{10,t} + 2}
\end{aligned}
$$

The delta-rule corresponds to the update rule of the Rescorla-Wagner model[34]. The leaky rule corresponds to the mean of an approximate posterior which is a Beta distribution whose parameters depend on the leaky counts of observations: $n_1 + 1$ and $n_0 + 1$ (see Ref.[52] for more details).

The output prediction value is equal to $\widehat{p}_t$ in the unigram case, and in the bigram case, to $\widehat{p}_{1|1,t}$ if $x_t = 1$ and $1 - \widehat{p}_{0|0,t}$ otherwise. The parameter α is a free parameter which is tuned to the observed environment during training with the same training data as the networks.

Training

For a given environment and a given type of agent among the network types and heuristic types, all the reported results are based on 20 agents, each sharing the same set of hyperparameters and initialized with a different random seed. During training, the parameters of a given agent were tuned to minimize the binary cross-entropy cost function (see equation (1)). During one iteration of training, the gradients of the cost function with respect to the parameters are computed on a subset of the training data (called a minibatch) using backpropagation through time and are used to update the parameters according to the selected training algorithm. The training algorithm was Adam[130] for the network types and stochastic gradient descent for the heuristic types.

For the unigram environment, the analyses reported in Fig. 2 to 4 were conducted after training on a common training dataset of 160 minibatches 20 sequences. For each of the two bigram environments, the analyses reported in Fig. 5 to 6 were conducted after training on a common training dataset (one per environment) of 400 minibatches of 20 sequences. These sizes were sufficient for the validation performance to converge before the end of training for all types of agents.

Parameters initialization. For all of the networks, the bias parameters are randomly initialized from a uniform distribution on $[-1/\sqrt{N}, +1/\sqrt{N}]$, and the weights w_{hp} are randomly initialized from a normal distribution with standard deviation $1/\sqrt{N}$ and mean 0. For all of the networks, the weights w_{xr}, w_{xz}, w_{xh} are randomly initialized from a normal distribution with standard deviation $\sigma_{0,x.}$ and mean 0, and the weights $w_{hr,ji}$, $w_{hz,ji}$, $w_{hh,ji}$ are randomly initialized from a normal distribution with standard deviation $\sigma_{0,h.}$ and mean 0 for all $j \neq i$ and $\mu_{0,h.,ii}$ for $j = i$. $\sigma_{0,x.}$, $\sigma_{0,h.}$, $\mu_{0,h.,ii}$ are hyperparameters that were optimized for a given environment, type of network, and number of units as detailed in the hyperparameter optimization section (the values resulting from this optimization are listed in Supplementary Table 1).

For the initialization of the parameter α in the heuristic solutions, a random value r is drawn from a log-uniform distribution on the interval [$10^{-2.5}$,$10^{-0.5}$], and the initial value of α is set to r in the delta-rule case or exp(-r) in the leaky rule case.

Hyperparameter optimization

Each type of agent had a specific set of hyperparameters to be optimized. For all network types, it included the initial learning rate of Adam η_0 and the initialization hyperparameters $\sigma_{0,x.}$, $\sigma_{0,h.}$. For the networks without lateral connections specifically, it also included $\mu_{0,h.,ii}$ (for those networks, setting it close to 1 can help avoid the vanishing gradient problem during training[131,132]); for the other networks, this was set to 0. For the heuristic types, it included only the learning rate of the stochastic gradient descent. A unique set of hyperparameter values was determined for each type of agent, each environment, and, for the network types, each number of units, through the optimization described next.

We used Bayesian optimization[133] with Gaussian processes and the upper confidence bound acquisition function to identify the best hyperparameters for each network architecture, environment, and number of units. During the optimization, combinations of hyperparameter values were iteratively sampled, each evaluated over 10 trials with different random seeds, for a total of 60 iterations (hence, 600 trials) for a given architecture, environment, and number of units. In each trial, one network was created, trained, and its cross-entropy was measured on independent test data. The training and test datasets used for the hyperparameter optimization procedure were not used in any other analyses. The training datasets contained respectively 160 and 400 minibatches of 20 sequences for the unigram and the bigram environment; the test datasets contained 200 sequences for each environment. We selected the combination of hyperparameter values corresponding to the iteration that led to the lowest mean test cross-entropy over the 10 trials. The selected values are listed in Supplementary Table 1.

For the heuristic types, we used random search from a log uniform distribution in the [10^{-6}, 10^{-1}] range over 80 trials to determine the optimal learning rate of the stochastic gradient descent. This led to selecting the value 3.10^{-3} for all heuristic types and all three environments.

Performance analyses

All agents were tested in the environment they were trained in (except for Supplementary Fig. 3 which tests cross-environment performance). We used a single test dataset per environment of 1,000 sequences independent of the training dataset. The log likelihood L of a given agent was measured from its predictions according to equation (1). The optimal log likelihood $L_{optimal}$ was measured from the predictions of the optimal solution for the given environment. The chance log likelihood L_{chance} was measured using a constant prediction of 0.5. To facilitate the interpretation of the results, the prediction performance of the agent was expressed as the % of optimal log likelihood, defined as $\frac{L - L_{chance}}{L_{optimal} - L_{chance}} \times 100$.

To test the statistical significance of a comparison of performance between two types of agents, we used a two-tailed two independent samples t-test with Welch's correction for unequal variances.

Analysis of the effective learning rate

The instantaneous effective learning rate of an agent that updates its prediction from p_t to p_{t+1} upon receiving when given as observation x_{t+1} is calculated as:

$$\alpha_{t+1} = \frac{p_{t+1} - p_t}{x_{t+1} - p_t}$$
$$\alpha_{t=0} = \frac{p_0 - 0.5}{x_0 - 0.5}$$ **(2)**

We call it “effective learning rate” because, had the agent been using a delta-rule algorithm, it would be equivalent to the learning rate of the delta-rule (as can be seen by rearranging the above formula into an update equation), and because it can be measured even if the agent uses another algorithm.

Readout analyses

The readout of a given quantity from the recurrent units of a network consists of a weighted sum of the activation values of each unit. To determine the weights of the readout for a given network, we ran a multiple linear regression using, as input variables, the activation of each recurrent unit at a given time step, $h_{i,t}$, and as target variable, the desired quantity calculated at the same time step. The regression was run on a training dataset of 900 sequences of 380 observations each (hence, 342,000 samples).

In the unigram environment, the precision readout was obtained using as desired quantity the log precision of the posterior distribution over the unigram variable calculated by the optimal solution as previously described, i.e. $\psi_t = -\log \sigma_t$, where σ_t is the standard deviation of the posterior distribution over p^{env}_{t+1}: $\sigma_t = \mathbb{SD}[p^{env}_{t+1}|x_0, \ldots, x_t]$ **(3)**.

In the bigram environment, the readout of the estimate of a given bigram variable was obtained using as desired quantity the log odds of the mean of the posterior distribution over that bigram variable calculated by the optimal solution, and the readout of the precision of that estimate was obtained using the log precision of that same posterior under the above definition of precision.

In Fig. 4b, to measure the accuracy of the readout from a given network, we calculated the Pearson correlation between the quantity read from the network and the optimal quantity on a test dataset of 100 sequences (hence, 38,000 samples), independent from any training dataset. To measure the Pearson correlation between the precision read and the effective learning rate, we used 300 out-of-sample sequences (hence, 114,000 samples).

In Fig. 5d, the log odds and log precision were transformed back into mean and standard deviation for visualization purposes.

Perturbation experiment to test precision-weighting

The perturbation experiment reported in Fig. 4c is designed to test the causal role of the precision read from a given network on its weighting of the next observation, measured through its effective learning rate. We performed this perturbation experiment on each of the 20 networks that were trained within each of the 4 architectures we considered. The causal instrument is a perturbation vector q that is added to the network's recurrent unit activations. The perturbation vector was randomly generated subject to the following constraints:

- $q \cdot w_{h\psi} = \delta\psi$ is the desired change in precision (we used 5 levels) that is read from the units' activities; it is computed by projecting the perturbation onto the weight vector of the precision readout ($w_{h\psi}$, $\cdot$ is the dot product);
- the perturbation q induces no change in the prediction of the network: $q \cdot w_{hp} = 0$, where w_{hp} is the weight vector of the output unit of the network;

- the perturbation has a constant intensity c across simulations, which we formalize as the norm of the perturbation: $\|q\| = c$.

We describe below the algorithm that we used to generate random perturbations q that satisfy these constraints. The idea is to decompose q into two components: both components leave the prediction unaffected, the first (q_ψ) is used to induce a controlled change in precision, the second (q_r) does not change the precision but is added to ensure a constant intensity of the perturbation across simulations.

1. To ensure no change in precision, we compute Q, the subspace of the activation space spanned by all vectors q that are orthogonal to the prediction weight vector w_{hp}, as the null space of w_{hp} (i.e. the orthogonal complement of the subspace spanned by w_{hp}, dimension N-1).
2. We compute q_ψ, the vector component of Q that affects precision, as the orthogonal projection of $w_{h\psi}$ onto Q .
3. We compute β_ψ , the coefficient to assign to q_ψ in the perturbation vector to produce the desired change in precision $\delta\psi$, as $\beta_\psi = \frac{\delta\psi}{\|q_\psi \cdot w_{h\psi}\|}$.
4. We compute R , the subspace spanned by all vector components of Q that do not affect precision, as the null space of q_ψ (dimension N-2). A perturbation vector in R therefore leaves both the prediction and the precision unchanged.
5. We draw a random unit vector q_r within R (by drawing from all N-2 components).
6. We compute β_r , the coefficient to assign to q_r in the perturbation vector so as to ensure that the final perturbation's norm equals c , as $\beta_r = \sqrt{c^2 - \beta_\psi^2 \|q_\psi\|^2}$.
7. We combine q_ψ and q_r into the final perturbation vector as $q = \beta_\psi q_\psi + \beta_r q_r$.

The experiment was run on a set of 1,000 sample time points randomly drawn from 300 sequences. First, the unperturbed learning rate was measured by running the network on all of the sequences. Second, for each sample time point, the network was run unperturbed up until that point, a perturbation vector was randomly generated for the desired change of precision and applied to the network at that point, then the perturbed network was run on the next time point and its perturbed learning rate was measured. This was repeated for each level of change in precision. Finally, for a given change in precision, the change in learning rate was calculated as the difference between the perturbed and the unperturbed learning rate.

For statistical analysis, we ran a one-tailed paired t-test to test whether the population's mean change in learning rate was higher at one level of precision change than at the next level of precision change. This was done for each of the four consecutive pairs of levels of change in precision.

Test of higher-level inference about changes

For a given network architecture, higher-level inference about changes was assessed by comparing the population of 20 networks trained in the environment with coupled change points to the population of 20 networks trained in the environment with independent change points.

In Fig. 6c, the change in unobserved bigram prediction for a given streak length *m* was computed as follows. First, prior sequences were generated and each network was run on each of the sequences. We generated initial sequences of 74 observations each with a probability of 0.2 for the 'observed' bigram

(which will render its repetition surprising) and a probability *p* for the 'unobserved' bigram equal to 0.2 or 0.8 (such probabilities, symmetric and substantially different from the default prior 0.5, should render a change in their inferred value detectable). We crossed all possibilities (0|0 or 1|1 as observed bigram, 0.2 or 0.8 for *p*) and generated 100 sequences for each (hence 400 sequences total). Second, at the end of each of these initial sequences, the prediction for the unobserved bigram, p_{before}, was queried by retrieving the output of the network after giving it as input '0' if the unobserved bigram was 0|0 or '1' otherwise. Third, the network was further presented with *m* repeated observations of the same value: '1' if the observed bigram was 1|1 or '0' otherwise. Finally, after this streak of repetition, the new prediction for the unobserved bigram, p_{after}, was queried (as before) and we measured its change with respect to the previous query, $|p_{after}-p_{before}|$. This procedure was repeated for *m* ranging from 2 and 75.

For statistics, we ran a one-tailed two independent samples t-test to test whether the mean change in unobserved bigram prediction of the population trained on coupled change points was higher than that of the population trained on independent change points.

Complexity analyses

The complexity analysis reported in Fig. 7 consisted in measuring, for each network architecture and each environment, the performance of optimally trained networks as a function of the number of units N. For optimal training, hyperparameter optimization was repeated at several values of N, for each type of network and each environment (the resulting values are listed in Supplementary Table 1). For the complexity analysis, a grid of equally spaced N values in logarithmic space between 1 and 45 was generated, and an additional value of 474 was included specifically for the networks without recurrent weight tuning so as to match their number of tuned parameters to that of an 11-unit gated recurrent network. For every value on this grid, 20 networks of a given architecture in a given environment were randomly initialized with the set of hyperparameter values that was determined to be optimal for the nearest neighboring N value in logarithmic space. The performance of these networks after training was evaluated using a new couple of training and test datasets per environment, each consisting of 400 minibatches of 20 sequences for training and 1,000 sequences for testing.

Statistics

To assess the variability between different agent solutions, we trained 20 agents for each type of agent and each environment. These agents have different random seeds (which changes their parameter initialization and how their training data is shuffled). Throughout the article, we report mean or median over these agents, and individual data points when possible or 95% confidence intervals (abbreviated as "CI") otherwise, as fully described in the text and figure legends. No statistical methods were used to pre-determine sample sizes but our sample sizes are similar to those reported in previous publications[75,134]. Data analysis was not performed blind to the conditions of the experiments. No data were excluded from the analyses. All statistical tests were two-tailed unless otherwise noted. The data distribution was assumed to be normal, but this was not formally tested. The specific details of each statistical analysis are reported directly in the text.

Reporting Summary

Further information on research design is available in the Nature Research Reporting Summary.

Acknowledgements

We thank Yair Lakretz for useful feedback, advice, and discussions throughout the project, Alexandre Pouget for his input when starting this project, and Charles Findling for comments on a previous version of the manuscript.

This project was funded by Inserm, ANR grant 18-CE37-0010-01 “CONFI LEARN” and ERC StG 947105 “NEURAL PROB” to F.M. C.F. received a PhD fellowship from ENS Paris Saclay.

Author contributions

C.F. and F.M. designed the research. C.F. performed the model simulations and analyses. C.F. and F.M. interpreted the results. C.F. and F.M. wrote the manuscript.

Competing interests

The authors declare no competing interests.

Code availability

The code to reproduce exhaustively the analyses of this paper will be publicly available on https://github.com/cedricfoucault and will be assigned a DOI on Zenodo at the time of publication. During the peer-review stage, reviewers and editors can download the code using a private URL. This code also enables to train new networks equipped with any number of units and generate Figures 2 to 6 with those networks.

Data availability

This paper presents no experimental data. All synthetic data are available in the code repository.

References

1. Sutton, R. S. & Barto, A. G. *Introduction to Reinforcement Learning*. (MIT Press, 1998).
2. Dolan, R. J. & Dayan, P. Goals and Habits in the Brain. *Neuron* **80**, 312–325 (2013).
3. Summerfield, C. & de Lange, F. P. Expectation in perceptual decision making: neural and computational mechanisms. *Nat. Rev. Neurosci.* **15**, 745–756 (2014).
4. Saffran, J. R., Aslin, R. N. & Newport, E. L. Statistical learning by 8-month-old infants. *Science* **274**, 1926–1928 (1996).
5. Dehaene, S., Meyniel, F., Wacongne, C., Wang, L. & Pallier, C. The Neural Representation of Sequences: From Transition Probabilities to Algebraic Patterns and Linguistic Trees. *Neuron* **88**, 2–19 (2015).
6. Sherman, B. E., Graves, K. N. & Turk-Browne, N. B. The prevalence and importance of statistical learning in human cognition and behavior. *Curr. Opin. Behav. Sci.* **32**, 15–20 (2020).
7. De Lange, F. P., Heilbron, M. & Kok, P. How do expectations shape perception? *Trends Cogn. Sci.* **22**, 764–779 (2018).
8. Gallistel, C. R., Krishan, M., Liu, Y., Miller, R. & Latham, P. E. The perception of probability. *Psychol. Rev.* **121**, 96–123 (2014).
9. Behrens, T. E. J., Woolrich, M. W., Walton, M. E. & Rushworth, M. F. S. Learning the value of information in an uncertain world. *Nat. Neurosci.* **10**, 1214–1221 (2007).
10. Soltani, A. & Izquierdo, A. Adaptive learning under expected and unexpected uncertainty. *Nat. Rev. Neurosci.* **20**, 635–644 (2019).
11. Sutton, R. Gain Adaptation Beats Least Squares? in *In Proceedings of the 7th Yale Workshop on Adaptive and Learning Systems* 161–166 (1992).
12. Fairhall, A. L., Lewen, G. D., Bialek, W. & de Ruyter van Steveninck, R. R. Efficiency and ambiguity in an adaptive neural code. *Nature* **412**, 787–792 (2001).
13. Wark, B., Fairhall, A. & Rieke, F. Timescales of Inference in Visual Adaptation. *Neuron* **61**, 750–761 (2009).
14. Sterling, P. Principles of allostasis: optimal design, predictive regulation, pathophysiology, and rational therapeutics. *Allostasis Homeost.*

Costs Physiol. Adapt. 17–64 (2004).
15. Pezzulo, G., Rigoli, F. & Friston, K. Active Inference, homeostatic regulation and adaptive behavioural control. *Prog. Neurobiol.* **134**, 17–35 (2015).
16. Berniker, M. & Kording, K. Estimating the sources of motor errors for adaptation and generalization. *Nat. Neurosci.* **11**, 1454–1461 (2008).
17. Wolpert, D. M., Ghahramani, Z. & Jordan, M. I. An internal model for sensorimotor integration. *Science* **269**, 1880–1882 (1995).
18. Iglesias, S. *et al.* Hierarchical prediction errors in midbrain and basal forebrain during sensory learning. *Neuron* **80**, 519–530 (2013).
19. Hauser, M. D., Newport, E. L. & Aslin, R. N. Segmentation of the speech stream in a non-human primate: statistical learning in cotton-top tamarins. *Cognition* **78**, B53–B64 (2001).
20. Rose, G. J. *et al.* Species-typical songs in white-crowned sparrows tutored with only phrase pairs. *Nature* **432**, 753–758 (2004).
21. Robinson, J. G. An Analysis of the Organization of Vocal Communication in the Titi Monkey Callicebus moloch. *Z. Für Tierpsychol.* **49**, 381–405 (1979).
22. Schapiro, A. C., Rogers, T. T., Cordova, N. I., Turk-Browne, N. B. & Botvinick, M. M. Neural representations of events arise from temporal community structure. *Nat. Neurosci.* **16**, 486–492 (2013).
23. Tauber, S., Navarro, D. J., Perfors, A. & Steyvers, M. Bayesian models of cognition revisited: Setting optimality aside and letting data drive psychological theory. *Psychol. Rev.* **124**, 410–441 (2017).
24. Ma, W. J. & Jazayeri, M. Neural coding of uncertainty and probability. *Annu. Rev. Neurosci.* **37**, 205–220 (2014).
25. Meyniel, F., Schlunegger, D. & Dehaene, S. The Sense of Confidence during Probabilistic Learning: A Normative Account. *PLOS Comput. Biol.* **11**, e1004305 (2015).
26. Heilbron, M. & Meyniel, F. Confidence resets reveal hierarchical adaptive learning in humans. *PLOS Comput. Biol.* **15**, e1006972 (2019).
27. Tenenbaum, J. B., Kemp, C., Griffiths, T. L. & Goodman, N. D. How to grow a mind: statistics, structure, and abstraction. *Science* **331**, 1279–1285 (2011).
28. Cooper, G. F. The computational complexity of probabilistic inference using Bayesian belief networks. *Artif. Intell.* **42**, 393–405 (1990).
29. Dagum, P. & Luby, M. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artif. Intell.* **60**, 141–153 (1993).
30. Ma, W. J., Beck, J. M., Latham, P. E. & Pouget, A. Bayesian inference with probabilistic population codes. *Nat. Neurosci.* **9**, 1432–1438 (2006).
31. Sahani, M. & Dayan, P. Doubly Distributional Population Codes: Simultaneous Representation of Uncertainty and Multiplicity. *Neural Comput.* **15**, 2255–2279 (2003).
32. Fiser, J., Berkes, P., Orbán, G. & Lengyel, M. Statistically optimal perception and learning: from behavior to neural representations. *Trends Cogn. Sci.* **14**, 119–130 (2010).
33. Todd, P. M. & Gigerenzer, G. Précis of Simple heuristics that make us smart. *Behav. Brain Sci.* **23**, 727–741 (2000).
34. Rescorla, Robert A. & Wagner, Allan R. A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. in *Classical conditioning II: Current research and theory* (eds. AH. et al.) 64–99 (New York Appleton-Century Crofts, 1972).
35. Schultz, W., Dayan, P. & Montague, P. R. A Neural Substrate of Prediction and Reward. *Science* **275**, 1593–1599 (1997).
36. Eshel, N., Tian, J. & Uchida, N. Opening the black box: dopamine, predictions, and learning. *Trends Cogn. Sci.* **17**, 430–431 (2013).
37. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
38. Saxe, A., Nelli, S. & Summerfield, C. If deep learning is the answer, what is the question? *Nat. Rev. Neurosci.* **22**, 55–67 (2021).
39. Nassar, M. R., Wilson, R. C., Heasly, B. & Gold, J. I. An Approximately Bayesian Delta-Rule Model Explains the Dynamics of Belief Updating in a Changing Environment. *J. Neurosci.* **30**, 12366–12378 (2010).
40. Nassar, M. R. *et al.* Rational regulation of learning dynamics by pupil-linked arousal systems. *Nat. Neurosci.* **15**, 1040–1046 (2012).
41. Elman, J. L. Distributed representations, simple recurrent networks, and grammatical structure. *Mach. Learn.* **7**, 195–225 (1991).
42. LeCun, Y. Predictive learning. *Proc Speech NIPS* (2016).
43. Cho, K. *et al.* Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* 1724–1734 (Association for Computational Linguistics, 2014). doi:10.3115/v1/D14-1179.
44. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS 2014 Workshop Deep Learn. Dec. 2014* (2014).
45. Elman, J. L. Finding Structure in Time. *Cogn. Sci.* **14**, 179–211 (1990).
46. Tanaka, G. *et al.* Recent advances in physical reservoir computing: A review. *Neural Netw.* **115**, 100–123 (2019).
47. Aston-Jones, G., Rajkowski, J. & Kubiak, P. Conditioned responses of monkey locus coeruleus neurons anticipate acquisition of discriminative behavior in a vigilance task. *Neuroscience* **80**, 697–715 (1997).
48. Kaliukhovich, D. A. & Vogels, R. Neurons in Macaque Inferior Temporal Cortex Show No Surprise Response to Deviants in Visual Oddball Sequences. *J. Neurosci.* **34**, 12801–12815 (2014).
49. Ulanovsky, N., Las, L., Farkas, D. & Nelken, I. Multiple time scales of adaptation in auditory cortex neurons. *J. Neurosci. Off. J. Soc. Neurosci.* **24**, 10440–10453 (2004).
50. Vinckier, F. *et al.* Confidence and psychosis: a neuro-computational account of contingency learning disruption by NMDA blockade. *Mol. Psychiatry* **21**, 946–955 (2016).

51. Yu, A. J. & Cohen, J. D. Sequential effects: Superstition or rational behavior? *Adv. Neural Inf. Process. Syst.* **21**, 1873–1880 (2008).
52. Meyniel, F., Maheu, M. & Dehaene, S. Human Inferences about Sequences: A Minimal Transition Probability Model. *PLOS Comput. Biol.* **12**, e1005260 (2016).
53. Gijsen, S., Grundei, M., Lange, R. T., Ostwald, D. & Blankenburg, F. Neural surprise in somatosensory Bayesian learning. *PLOS Comput. Biol.* **17**, e1008068 (2021).
54. Findling, C., Chopin, N. & Koechlin, E. Imprecise neural computations as a source of adaptive behaviour in volatile environments. *Nat. Hum. Behav.* **5**, 99–112 (2021).
55. Boldt, A., Blundell, C. & De Martino, B. Confidence modulates exploration and exploitation in value-based learning. *Neurosci. Conscious.* **2019**, (2019).
56. McGuire, J. T., Nassar, M. R., Gold, J. I. & Kable, J. W. Functionally Dissociable Influences on Learning Rate in a Dynamic Environment. *Neuron* **84**, 870–881 (2014).
57. Bornstein, A. M. & Daw, N. D. Cortical and Hippocampal Correlates of Deliberation during Model-Based Decisions for Rewards in Humans. *PLOS Comput. Biol.* **9**, e1003387 (2013).
58. Purcell, B. A. & Kiani, R. Hierarchical decision processes that operate over distinct timescales underlie choice and changes in strategy. *Proc. Natl. Acad. Sci.* **113**, E4531–E4540 (2016).
59. Bill, J., Pailian, H., Gershman, S. J. & Drugowitsch, J. Hierarchical structure is employed by humans during visual motion perception. *Proc. Natl. Acad. Sci.* **117**, 24581–24589 (2020).
60. Knill, D. C. & Pouget, A. The Bayesian brain: the role of uncertainty in neural coding and computation. *Trends Neurosci.* **27**, 712–719 (2004).
61. Bowers, J. S. & Davis, C. J. Bayesian just-so stories in psychology and neuroscience. *Psychol. Bull.* **138**, 389–414 (2012).
62. Griffiths, T. L., Chater, N., Norris, D. & Pouget, A. How the Bayesians got their beliefs (and what those beliefs actually are): Comment on Bowers and Davis (2012). *Psychol. Bull.* **138**, 415–422 (2012).
63. Rahnev, D. & Denison, R. N. Suboptimality in perceptual decision making. *Behav. Brain Sci.* **41**, (2018).
64. Lee, T. S. & Mumford, D. Hierarchical Bayesian inference in the visual cortex. *J. Opt. Soc. Am. A Opt. Image Sci. Vis.* **20**, 1434–1448 (2003).
65. Rao, R. P. & Ballard, D. H. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nat. Neurosci.* **2**, 79–87 (1999).
66. Sanborn, A. N. & Chater, N. Bayesian Brains without Probabilities. *Trends Cogn. Sci.* **20**, 883–893 (2016).
67. Chater, N., Tenenbaum, J. B. & Yuille, A. Probabilistic models of cognition: conceptual foundations. *Trends Cogn. Sci.* **10**, 287–291 (2006).
68. Findling, C., Skvortsova, V., Dromnelle, R., Palminteri, S. & Wyart, V. Computational noise in reward-guided learning drives behavioral variability in volatile environments. *Nat. Neurosci.* **22**, 2066–2077 (2019).
69. Wyart, V. & Koechlin, E. Choice variability and suboptimality in uncertain environments. *Curr. Opin. Behav. Sci.* **11**, 109–115 (2016).
70. Simon, H. A. A Behavioral Model of Rational Choice. *Q. J. Econ.* **69**, 99–118 (1955).
71. Simon, H. A. Theories of bounded rationality. *Decis. Organ.* **1**, 161–176 (1972).
72. Ma, W. J. Signal detection theory, uncertainty, and Poisson-like population codes. *Vision Res.* **50**, 2308–2319 (2010).
73. Deroy, O., Spence, C. & Noppeney, U. Metacognition in Multisensory Perception. *Trends Cogn. Sci.* **20**, 736–747 (2016).
74. Courville, A. C., Daw, N. D. & Touretzky, D. S. Bayesian theories of conditioning in a changing world. *Trends Cogn. Sci.* **10**, 294–300 (2006).
75. Yang, G. R., Joglekar, M. R., Song, H. F., Newsome, W. T. & Wang, X.-J. Task representations in neural networks trained to perform many cognitive tasks. *Nat. Neurosci.* **22**, 297–306 (2019).
76. Zhang, Z., Cheng, H. & Yang, T. A recurrent neural network framework for flexible and adaptive decision making based on sequence learning. *PLOS Comput. Biol.* **16**, e1008342 (2020).
77. Wang, J. X. *et al.* Prefrontal cortex as a meta-reinforcement learning system. *Nat. Neurosci.* **21**, 860–868 (2018).
78. Jazayeri, M. & Movshon, J. A. Optimal representation of sensory information by neural populations. *Nat. Neurosci.* **9**, 690–696 (2006).
79. Fusi, S., Asaad, W. F., Miller, E. K. & Wang, X.-J. A neural circuit model of flexible sensorimotor mapping: learning and forgetting on multiple timescales. *Neuron* **54**, 319–333 (2007).
80. Soltani, A. & Wang, X.-J. Synaptic computation underlying probabilistic inference. *Nat. Neurosci.* **13**, 112–119 (2010).
81. Echeveste, R., Aitchison, L., Hennequin, G. & Lengyel, M. Cortical-like dynamics in recurrent circuits optimized for sampling-based probabilistic inference. *Nat. Neurosci.* **23**, 1138–1149 (2020).
82. Pecevski, D., Buesing, L. & Maass, W. Probabilistic Inference in General Graphical Models through Sampling in Stochastic Networks of Spiking Neurons. *PLOS Comput. Biol.* **7**, e1002294 (2011).
83. Iigaya, K. Adaptive learning and decision-making under uncertainty by metaplastic synapses guided by a surprise detection system. *eLife* **5**, (2016).
84. Farashahi, S. *et al.* Metaplasticity as a Neural Substrate for Adaptive Learning and Choice under Uncertainty. *Neuron* **94**, 401-414.e6 (2017).
85. Fusi, S., Drew, P. J. & Abbott, L. F. Cascade Models of Synaptically Stored Memories. *Neuron* **45**, 599–611 (2005).
86. Barron, A. R. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Inf. Theory* **39**, 930–945 (1993).

87. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Syst.* **2**, 303–314 (1989).
88. Schäfer, A. M. & Zimmermann, H. G. Recurrent Neural Networks Are Universal Approximators. in *Artificial Neural Networks – ICANN 2006* (eds. Kollias, S. D., Stafylopatis, A., Duch, W. & Oja, E.) 632–640 (Springer Berlin Heidelberg, 2006).
89. Douglas, R. J. & Martin, K. A. C. Recurrent neuronal circuits in the neocortex. *Curr. Biol.* **17**, R496–R500 (2007).
90. Hunt, L. T. & Hayden, B. Y. A distributed, hierarchical and recurrent framework for reward-based choice. *Nat. Rev. Neurosci.* **18**, 172–182 (2017).
91. Yang, G. R., Murray, J. D. & Wang, X.-J. A dendritic disinhibitory circuit mechanism for pathway-specific gating. *Nat. Commun.* **7**, 12815 (2016).
92. Costa, R., Assael, I. A., Shillingford, B., de Freitas, N. & Vogels, Ti. Cortical microcircuits as gated-recurrent neural networks. *Adv. Neural Inf. Process. Syst.* **30**, (2017).
93. O'Reilly, R. C. Biologically Based Computational Models of High-Level Cognition. *Science* **314**, 91–94 (2006).
94. O'Reilly, R. C. & Frank, M. J. Making Working Memory Work: A Computational Model of Learning in the Prefrontal Cortex and Basal Ganglia. *Neural Comput.* **18**, 283–328 (2006).
95. Yamakawa, H. Attentional Reinforcement Learning in the Brain. *New Gener. Comput.* **38**, 49–64 (2020).
96. Klimesch, W. EEG alpha and theta oscillations reflect cognitive and memory performance: a review and analysis. *Brain Res. Rev.* **29**, 169–195 (1999).
97. Busch, N. A., Dubois, J. & VanRullen, R. The Phase of Ongoing EEG Oscillations Predicts Visual Perception. *J. Neurosci.* **29**, 7869–7876 (2009).
98. Mathewson, K. E., Gratton, G., Fabiani, M., Beck, D. M. & Ro, T. To See or Not to See: Prestimulus α Phase Predicts Visual Awareness. *J. Neurosci.* **29**, 2725–2732 (2009).
99. Hipp, J. F., Engel, A. K. & Siegel, M. Oscillatory Synchronization in Large-Scale Cortical Networks Predicts Perception. *Neuron* **69**, 387–396 (2011).
100. Baumgarten, T. J., Schnitzler, A. & Lange, J. Prestimulus Alpha Power Influences Tactile Temporal Perceptual Discrimination and Confidence in Decisions. *Cereb. Cortex* **26**, 891–903 (2016).
101. Iemi, L. *et al.* Multiple mechanisms link prestimulus neural oscillations to sensory responses. *eLife* **8**, e43620 (2019).
102. Jensen, O. & Mazaheri, A. Shaping Functional Architecture by Oscillatory Alpha Activity: Gating by Inhibition. *Front. Hum. Neurosci.* **4**, (2010).
103. Klimesch, W., Sauseng, P. & Hanslmayr, S. EEG alpha oscillations: The inhibition–timing hypothesis. *Brain Res. Rev.* **53**, 63–88 (2007).
104. Hahn, G., Ponce-Alvarez, A., Deco, G., Aertsen, A. & Kumar, A. Portraits of communication in neuronal networks. *Nat. Rev. Neurosci.* **20**, 117–127 (2019).
105. Aston-Jones, G. & Cohen, J. D. An integrative theory of locus coeruleus-norepinephrine function: adaptive gain and optimal performance. *Annu. Rev. Neurosci.* **28**, 403–450 (2005).
106. Servan-Schreiber, D., Printz, H. & Cohen, J. D. A network model of catecholamine effects: gain, signal-to-noise ratio, and behavior. *Science* **249**, 892–895 (1990).
107. Salgado, H., Treviño, M. & Atzori, M. Layer- and area-specific actions of norepinephrine on cortical synaptic transmission. *Brain Res.* **1641**, 163–176 (2016).
108. Moyer, J. T., Wolf, J. A. & Finkel, L. H. Effects of Dopaminergic Modulation on the Integrative Properties of the Ventral Striatal Medium Spiny Neuron. *J. Neurophysiol.* **98**, 3731–3748 (2007).
109. Thurley, K., Senn, W. & Lüscher, H.-R. Dopamine Increases the Gain of the Input-Output Response of Rat Prefrontal Pyramidal Neurons. *J. Neurophysiol.* **99**, 2985–2997 (2008).
110. Stalter, M., Westendorff, S. & Nieder, A. Dopamine Gates Visual Signals in Monkey Prefrontal Cortex Neurons. *Cell Rep.* **30**, 164-172.e4 (2020).
111. Herrero, J. L. *et al.* Acetylcholine contributes through muscarinic receptors to attentional modulation in V1. *Nature* **454**, 1110–1114 (2008).
112. Gil, Z., Connors, B. W. & Amitai, Y. Differential Regulation of Neocortical Synapses by Neuromodulators and Activity. *Neuron* **19**, 679–686 (1997).
113. Thiele, A. & Bellgrove, M. A. Neuromodulation of Attention. *Neuron* **97**, 769–785 (2018).
114. Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J. & Hinton, G. Backpropagation and the brain. *Nat. Rev. Neurosci.* **21**, 335–346 (2020).
115. Zador, A. M. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nat. Commun.* **10**, 3770 (2019).
116. Den Ouden, H. E., Kok, P. & De Lange, F. P. How Prediction Errors Shape Perception, Attention, and Motivation. *Front. Psychol.* **3**, (2012).
117. Maheu, M., Dehaene, S. & Meyniel, F. Brain signatures of a multiscale process of sequence learning in humans. *eLife* **8**, e41541 (2019).
118. O'Reilly, R. C., Russin, J. L., Zolfaghar, M. & Rohrlich, J. Deep Predictive Learning in Neocortex and Pulvinar. *J. Cogn. Neurosci.*

1–39 (2021) doi:10.1162/jocn_a_01708.
119. Peterson, C. R. & Beach, L. R. Man as an intuitive statistician. *Psychol. Bull.* **68**, 29–46 (1967).
120. Khaw, M. W., Stevens, L. & Woodford, M. Individual differences in the perception of probability. *PLOS Comput. Biol.* **17**, e1008871 (2021).
121. Payzan-LeNestour, E., Dunne, S., Bossaerts, P. & O'Doherty, J. P. The Neural Representation of Unexpected Uncertainty during Value-Based Decision Making. *Neuron* **79**, 191–201 (2013).
122. Friston, K. *et al.* Active inference and epistemic value. *Cogn. Neurosci.* **6**, 187–214 (2015).
123. Amini, A., Schwarting, W., Soleimany, A. & Rus, D. Deep Evidential Regression. *Adv. Neural Inf. Process. Syst.* **33**, 14927–14937 (2020).
124. Haxby, J. V., Connolly, A. C. & Guntupalli, J. S. Decoding Neural Representational Spaces Using Multivariate Pattern Analysis. *Annu. Rev. Neurosci.* **37**, 435–456 (2014).
125. Kriegeskorte, N. & Diedrichsen, J. Peeling the Onion of Brain Representations. *Annu. Rev. Neurosci.* **42**, 407–432 (2019).
126. Meyniel, F. & Dehaene, S. Brain networks for confidence weighting and hierarchical inference during probabilistic learning. *Proc. Natl. Acad. Sci.* **114**, E3859–E3868 (2017).
127. Meyniel, F. Brain dynamics for confidence-weighted learning. *PLOS Comput. Biol.* **16**, e1007935 (2020).
128. Bach, D. R., Hulme, O., Penny, W. D. & Dolan, R. J. The Known Unknowns: Neural Representation of Second-Order Uncertainty, and Ambiguity. *J. Neurosci.* **31**, 4811–4820 (2011).
129. Tomov, M. S., Truong, V. Q., Hundia, R. A. & Gershman, S. J. Dissociable neural correlates of uncertainty underlie different exploration strategies. *Nat. Commun.* **11**, 2371 (2020).
130. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (eds. Bengio, Y. & LeCun, Y.) (2015).
131. Bengio, Y., Simard, P. & Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **5**, 157–166 (1994).
132. Sutskever, I., Martens, J., Dahl, G. & Hinton, G. On the importance of initialization and momentum in deep learning. in *International Conference on Machine Learning* 1139–1147 (PMLR, 2013).
133. Agnihotri, A. & Batra, N. Exploring Bayesian Optimization. *Distill* **5**, e26 (2020).
134. Masse, N. Y., Yang, G. R., Song, H. F., Wang, X.-J. & Freedman, D. J. Circuit mechanisms for the maintenance and manipulation of information in working memory. *Nat. Neurosci.* **22**, 1159–1167 (2019).

Figures

Figures start on next page.

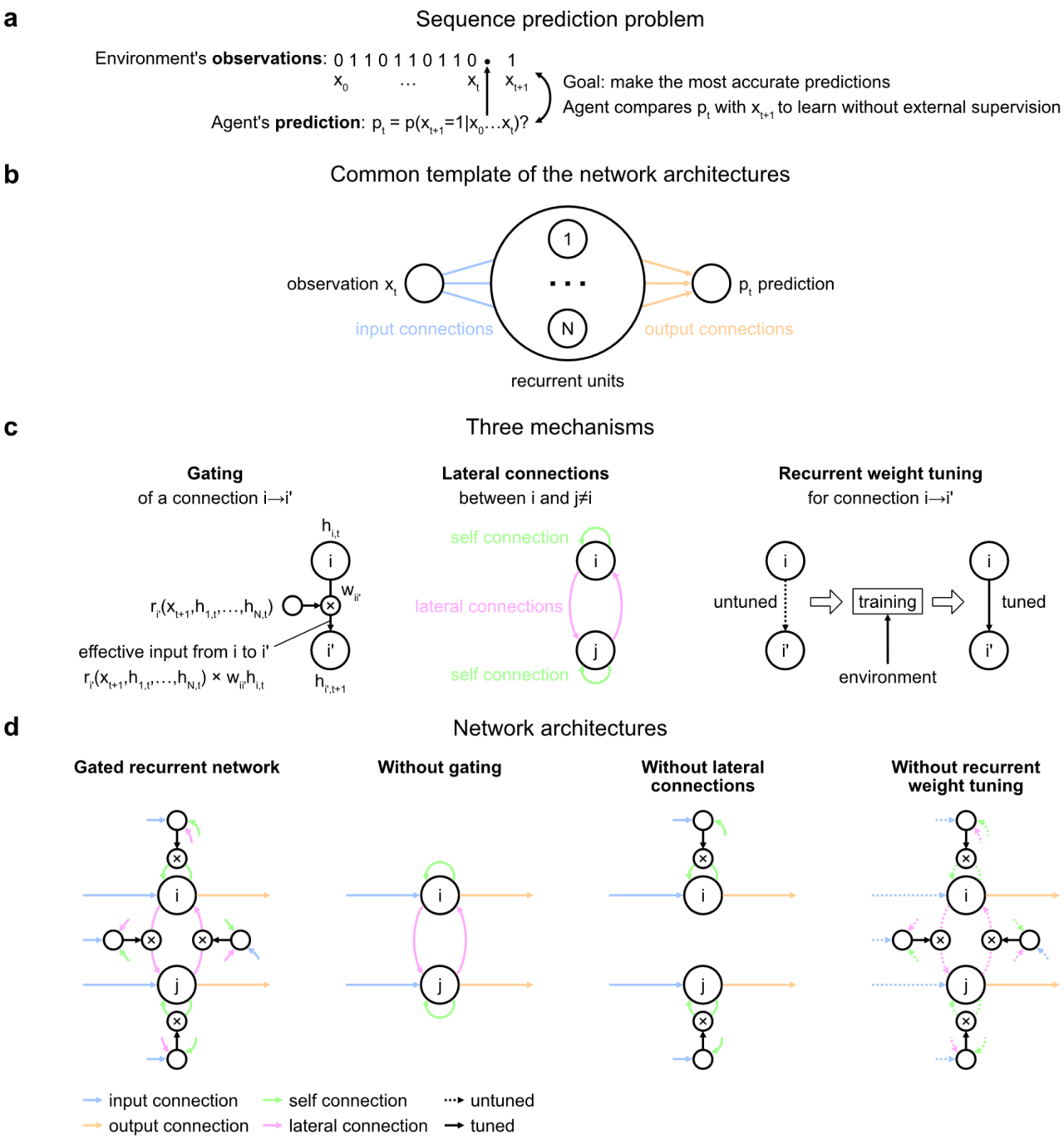


Fig. 1 | Problem to solve and network architectures. a, Sequence prediction problem. At time t, the environment generates a binary observation x_t. The agent receives it and returns a prediction p_t: its estimate of the probability that the next observation will be 1 given the observations collected so far. The agent's goal is to make the most accurate predictions possible. The agent can measure its accuracy by comparing its prediction p_t with the actual value observed at the next time step x_{t+1}, allowing it to learn from the observations without any external supervision. **b**, Common three-layer template of the recurrent neural network architectures. Input connections transmit the observation to the recurrent units and output connections allow the prediction to be read from the recurrent units. **c**, Three key mechanisms of recurrent neural network architectures. Gating allows multiplicative interaction between activities. Lateral connections allow the activities of different recurrent units i and j to interact. Recurrent weight tuning allows the connection weights of recurrent units to be adjusted to the training environment. i' may be equal to i. **d**, The gated recurrent architecture includes all three mechanisms: gating, lateral connections, and recurrent weight tuning. Each alternative architecture includes all but one of the three mechanisms.

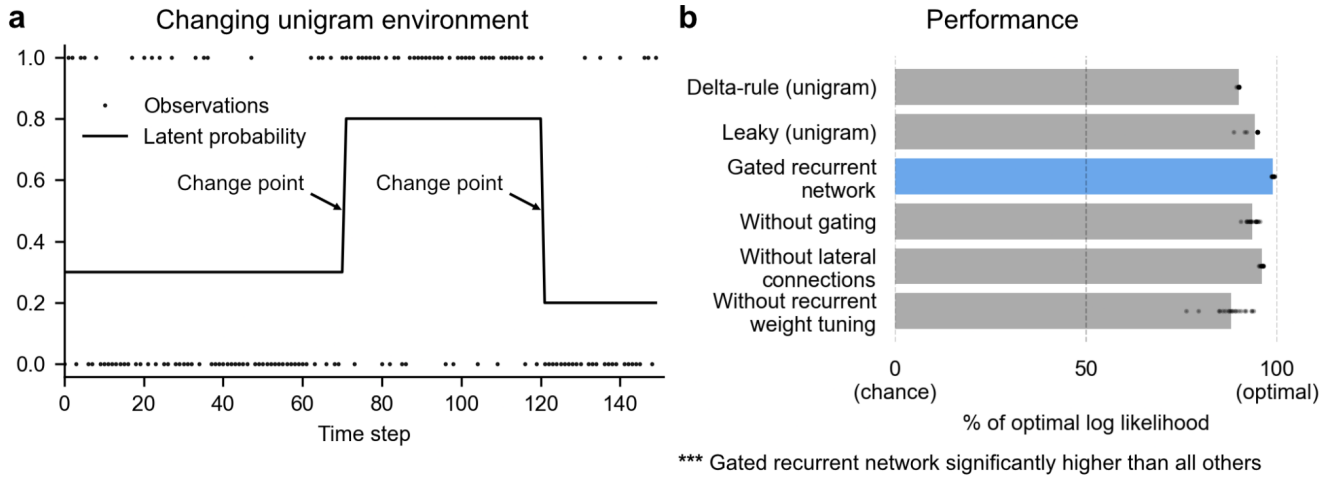


Fig. 2 | Gated recurrent networks perform quasi-optimally in the face of changes in latent probabilities. a, Sample sequence of observations (dots) and latent unigram probability (line) generated in the changing unigram environment. At each time step, a binary observation is randomly generated based on the latent unigram probability, and a change point can occur with a fixed probability, suddenly changing the unigram probability to a new value uniformly drawn in [0,1]. **b**, Prediction performance in the changing unigram environment. For each type of agent, 20 trained agents (trained with different random seeds) were tested (dots: agents; bars: average). Their prediction performance was measured as the % of optimal log likelihood (0% being chance performance and 100% optimal performance, see equation (1) for the log likelihood) and averaged over observations and sequences. The gated recurrent network significantly outperformed every other type of agent ($p < 0.001$, two-tailed two independent samples t-test with Welch's correction for unequal variances).

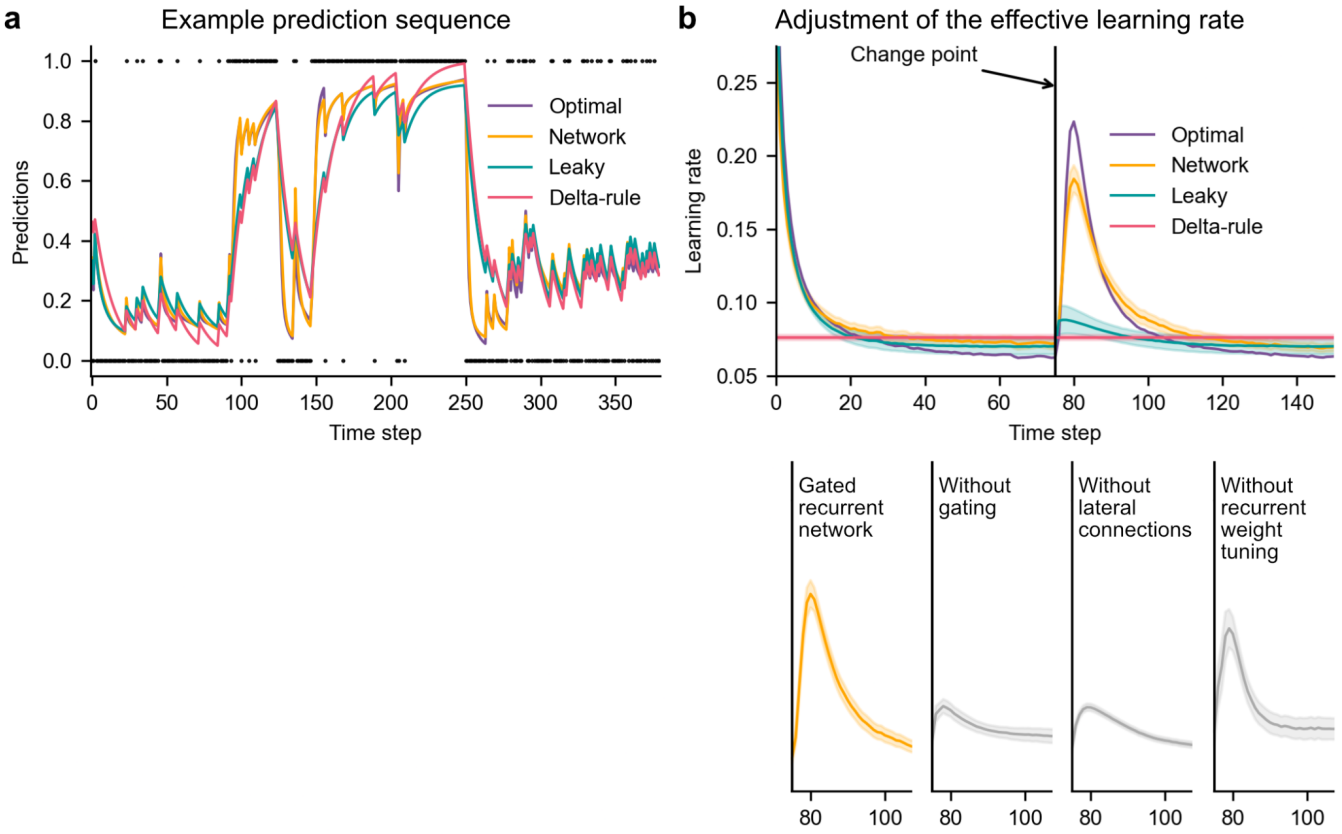


Fig. 3 | Gated recurrent but not alternative networks adjust their moment-by-moment effective learning rate around changes like the optimal agent. a, Example prediction sequence illustrating the prediction updates of different types of agents. Within each type of agent, the agent (out of 20) yielding median performance in Fig. 2b was selected for illustration purposes. Dots are observations, lines are predictions. **b**, Moment-by-moment effective learning rate of each type of agent. 20 trained agents of each type were tested on 10,000 sequences whose change points were locked at the same time steps, for illustration purposes. The moment-by-moment effective learning rate was measured as the ratio of prediction update to prediction error (see Methods, equation (2)), and averaged over sequences. Lines and bands show the mean and the 95% confidence interval of the mean.

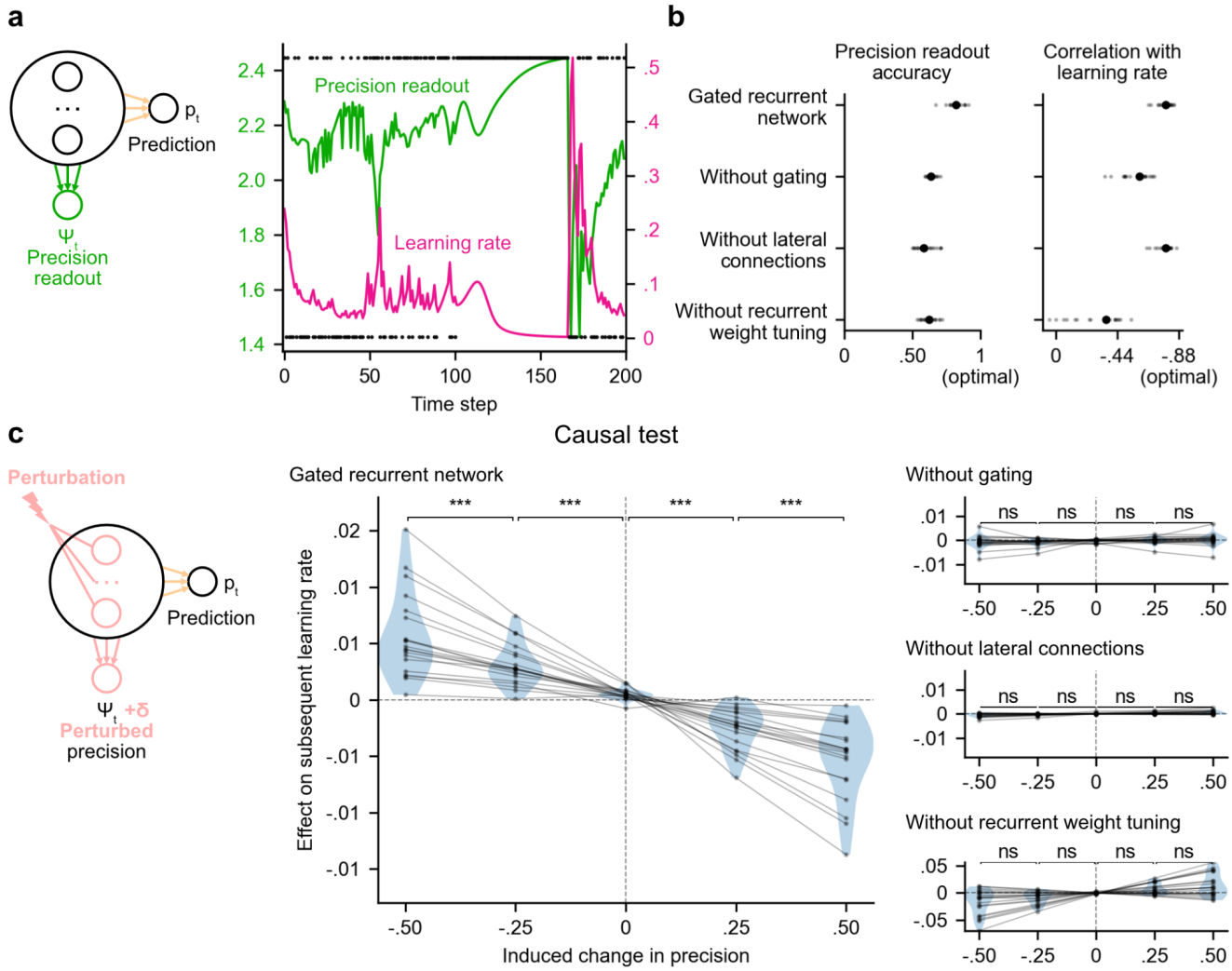


Fig. 4 | Precision-weighting causally determines the adjustment of the effective learning rate in gated recurrent networks only. a, Schematic of the readout of precision from the recurrent units of a network, and sample sequence illustrating the anti-correlation between the moment-by-moment precision read from a gated recurrent network and its effective learning rate (using the network that yielded median readout accuracy on Fig. 4b out of the 20 networks). The readout was obtained by fitting a multiple linear regression from the recurrent activity to the log precision of the optimal posterior distribution on the prediction (see equation (3)). **b**, Precision readout accuracy (left plot) and Pearson correlation between the precision read and the subsequent effective learning rate (right plot). The readout accuracy was calculated as the Pearson correlation between the precision read from the network and the optimal precision. The optimal correlation between precision and subsequent effective learning rate was calculated from the optimal agent's own precision and learning rate, on the same sequences as the networks. Large and small dots show the individual and median values, respectively. **c**, Causal test of a network's precision on its effective learning rate. The recurrent activity was perturbed to induce a controlled change in the precision read without changing the prediction on a given time step. Prediction error on the next time step is thus equal with and without perturbation. We measured the perturbation's effect on the effective learning rate as the difference in learning rate "with perturbation" minus "without perturbation" at the next time step. Each dot (and joining line) corresponds to one network. ***: $p < 0.001$, n.s.: $p > 0.05$ (one-tailed paired t-test).

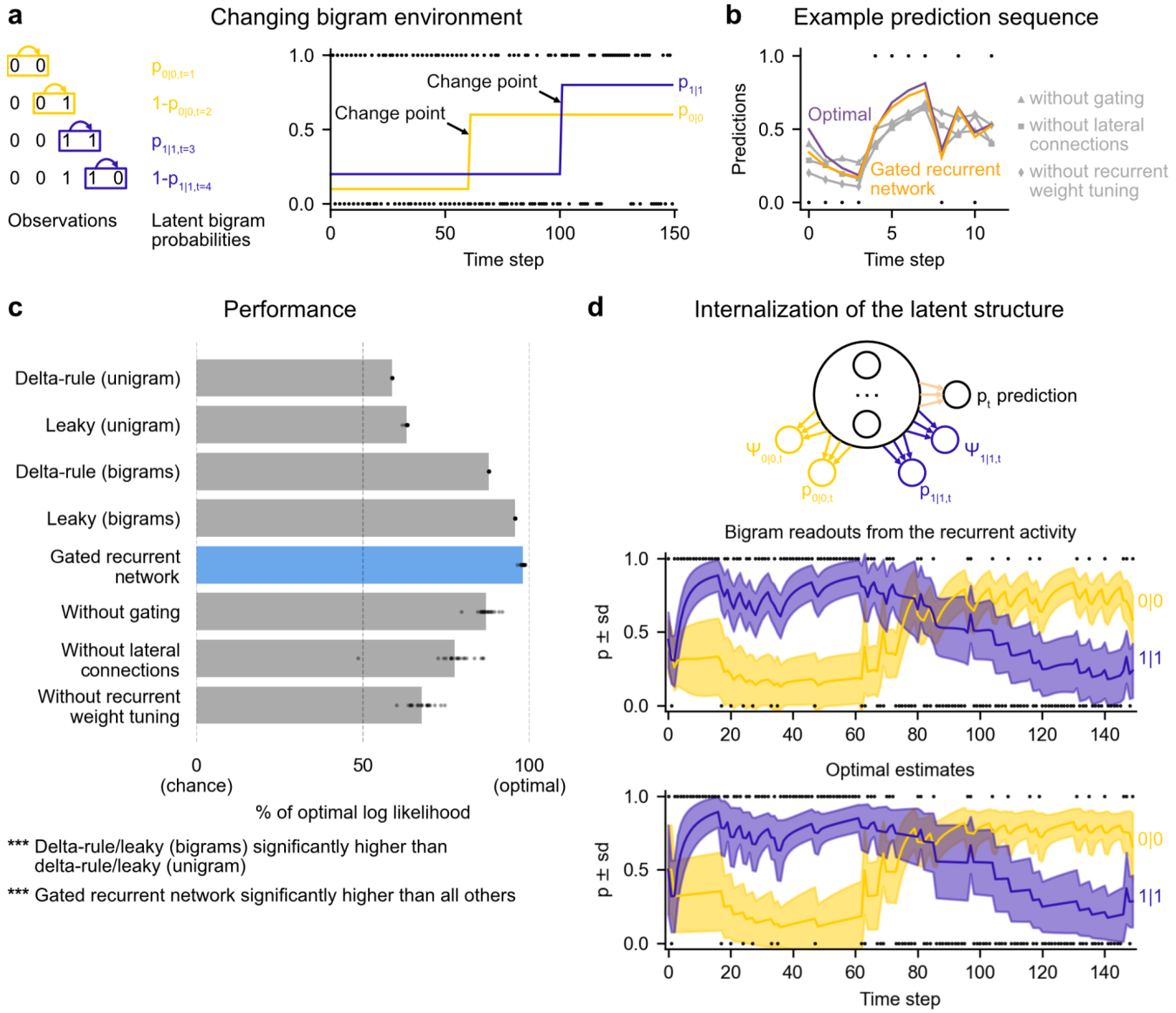


Fig. 5 | Gated recurrent networks correctly leverage and internalize the latent bigram structure. a, Schematic of the changing bigram environment's latent probabilities (left) and sample generated sequence (right, dots: observations, lines: latent bigram probabilities). At each time step, a binary observation is randomly generated according to the relevant latent bigram probability, $p_{0|0}$ or $p_{1|1}$ depending on the previous observation. $p_{0|0}$ denotes the probability of occurrence of a 0 after a 0 and $p_{1|1}$ that of a 1 after a 1 (note that $p_{1|0}=1-p_{0|0}$ and $p_{0|1}=1-p_{1|1}$). At any time step, each of the two bigram probabilities can suddenly change to a new value uniformly drawn in [0,1], randomly with a fixed probability and independently from each other. **b**, Example prediction sequence illustrating each network's ability or inability to change prediction according to the local context, compared to the optimal prediction (dots: observations, lines: predictions). **c**, Prediction performance of each type of agent in the changing bigram environment. 20 new agents of each type were trained and tested as in Fig. 2b but now in the changing bigram environment (dots: agents; bars: average). The gated recurrent network significantly outperformed every other type of agent ($p < 0.001$, two-tailed two independent samples t-test with Welch's correction for unequal variances). **d**, Internalization of the latent structure in the gated recurrent network: the two bigram probabilities are simultaneously represented, and their readouts closely follow the optimal estimates (top and bottom plots show an out-of-sample sequence). The readouts were obtained through linear regression from the recurrent activity to four estimates separately: the log odds of the mean and the log precision of the optimal posterior distribution on $p_{0|0}$ and $p_{1|1}$. In b and d, the networks (out of 20) yielding median performance were selected for illustration purposes.

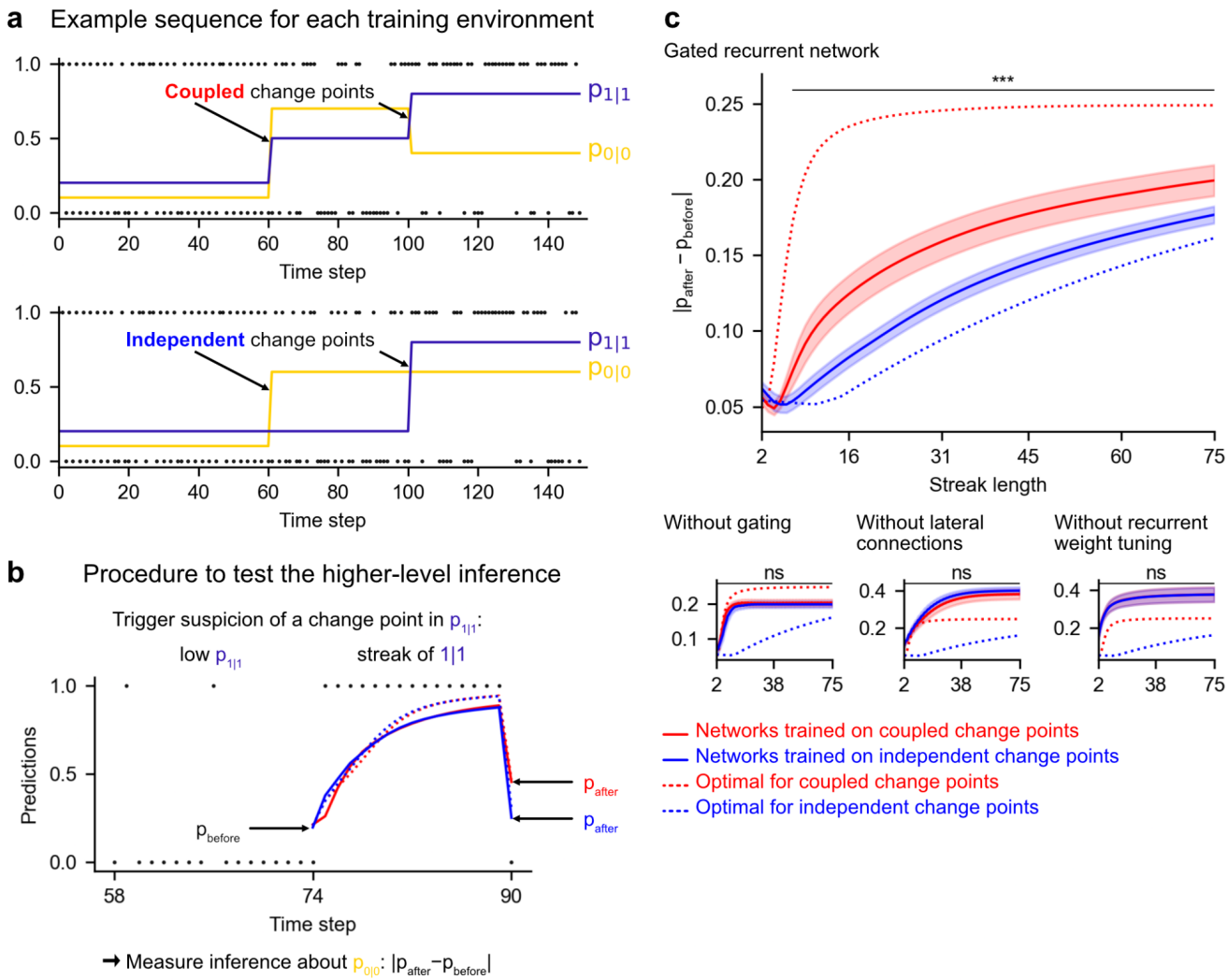


Fig. 6 | Gated recurrent but not alternative networks leverage a higher-level structure, distinguishing the case where change points are coupled vs. independent. Procedure to test the higher-level inference: **a**, For each network architecture, 20 networks were trained on sequences where the change points of the two latent bigram probabilities are coupled and 20 other networks were trained on sequences where they are independent (the plots show an example training sequence for each case); **b**, The networks were then tested on sequences designed to trigger the suspicion of a change point in one bigram probability and measure their inference about the other bigram probability: $|p_{after}-p_{before}|$ should be larger when the agent assumes change points to be coupled rather than independent. The plot shows an example test sequence. Red, blue, solid, and dashed lines: as in c, except that only the gated recurrent network (out of 20) yielding median performance is shown for illustration purposes. **c**, Change in prediction about the unobserved bigram probability of the networks trained on coupled change points (red) and independent change points (blue) for each network architecture, averaged over sequences. Solid lines and bands show the mean and the 95% confidence interval of the mean over networks. Dotted lines show the corresponding values of the optimal agent for the two cases. Only the gated recurrent architecture yields a significant difference between networks trained on coupled vs. independent change points (one-tailed two independent samples t-test, ***: $p < 0.001$, n.s.: $p > 0.05$).

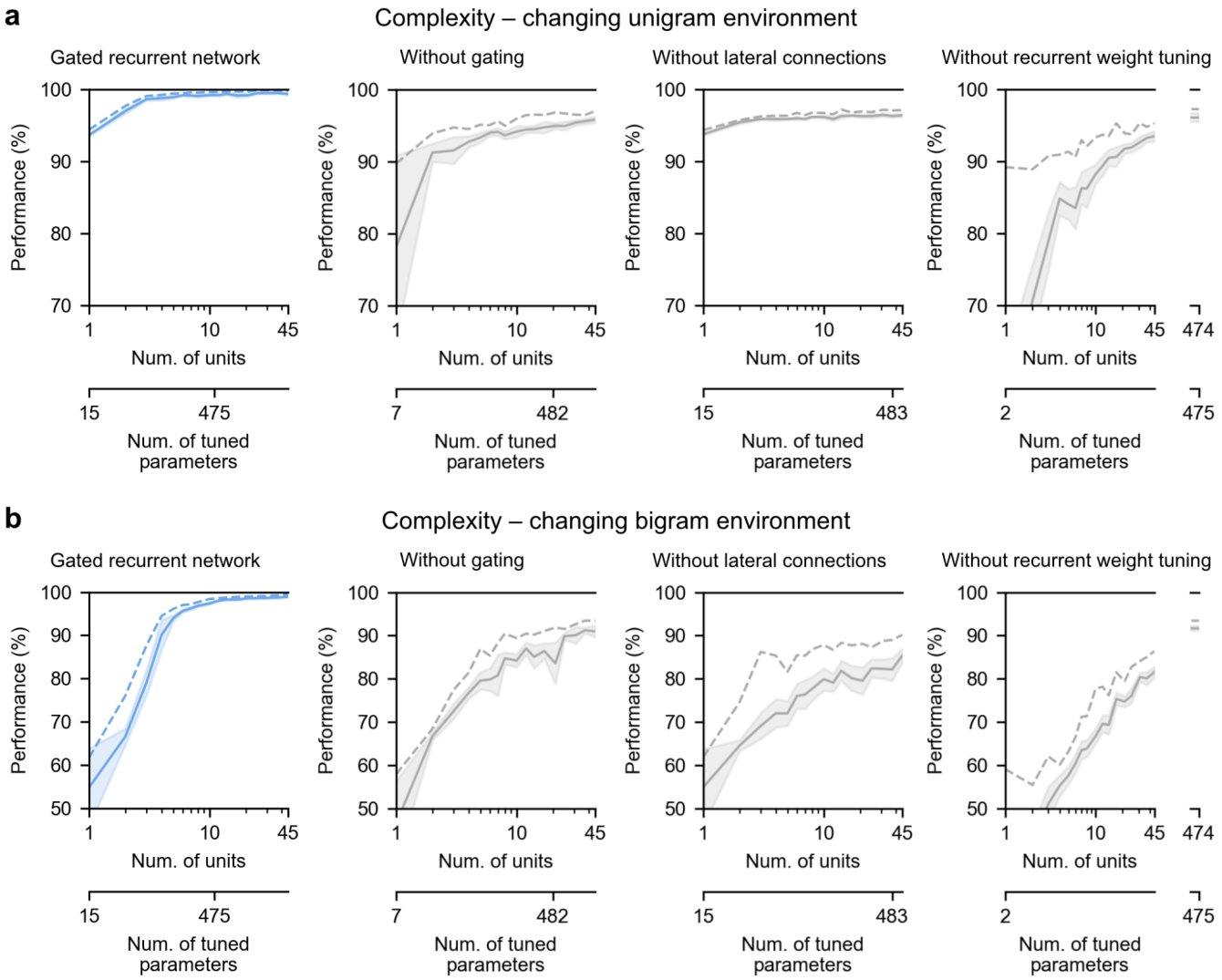


Fig. 7 | Low-complexity solutions are uniquely enabled by the combination of gating, lateral connections, and recurrent weight tuning. a and **b**, Prediction performance of each network architecture in the changing unigram environment and the changing bigram environment, respectively, as a function of the number of recurrent units (i.e. space complexity) of the network. For each network architecture and each number of units, 20 networks were trained using hyperparameters that had been optimized prior to training, and prediction performance was measured as the % of optimal log likelihood on new test sequences. Solid lines, bands, and dotted lines show the mean, 95% confidence interval of the mean, and maximum performance, respectively. At the maximum displayed number of units, all of the alternative architectures exceed the complexity of the 11-unit gated recurrent network shown on the left and in previous Figures, both in terms of the number of units and the number of tuned parameters (indicated on the twin x-axes), but none of them reach its performance.

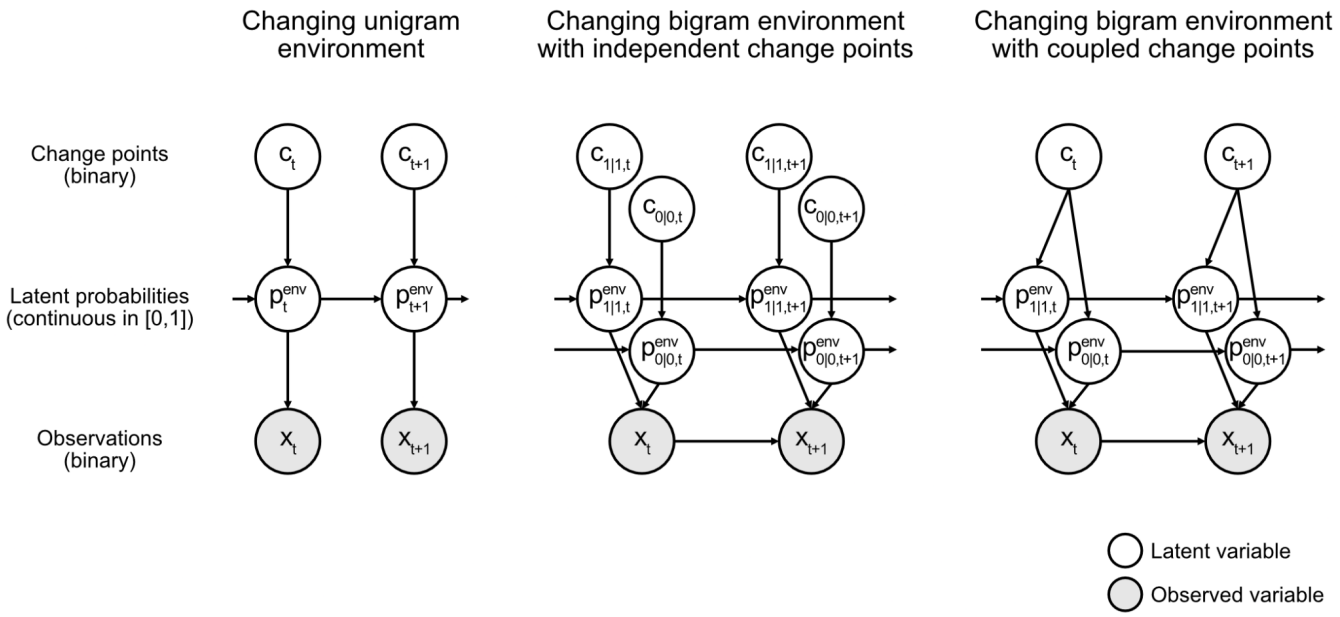
SUPPLEMENTARY INFORMATION

Gated recurrence enables simple and accurate sequence prediction in stochastic, changing, and structured environments

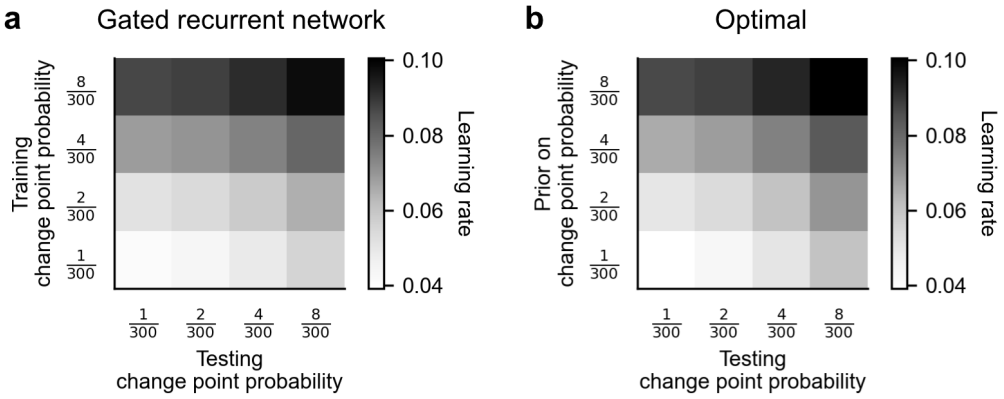
Cedric Foucault[1,2,*] and Florent Meyniel[1,*]

1. Cognitive Neuroimaging Unit, INSERM, CEA, Université Paris-Saclay, NeuroSpin center, 91191 Gif/Yvette, France.
2. Sorbonne Université, Collège Doctoral, F-75005 Paris, France.

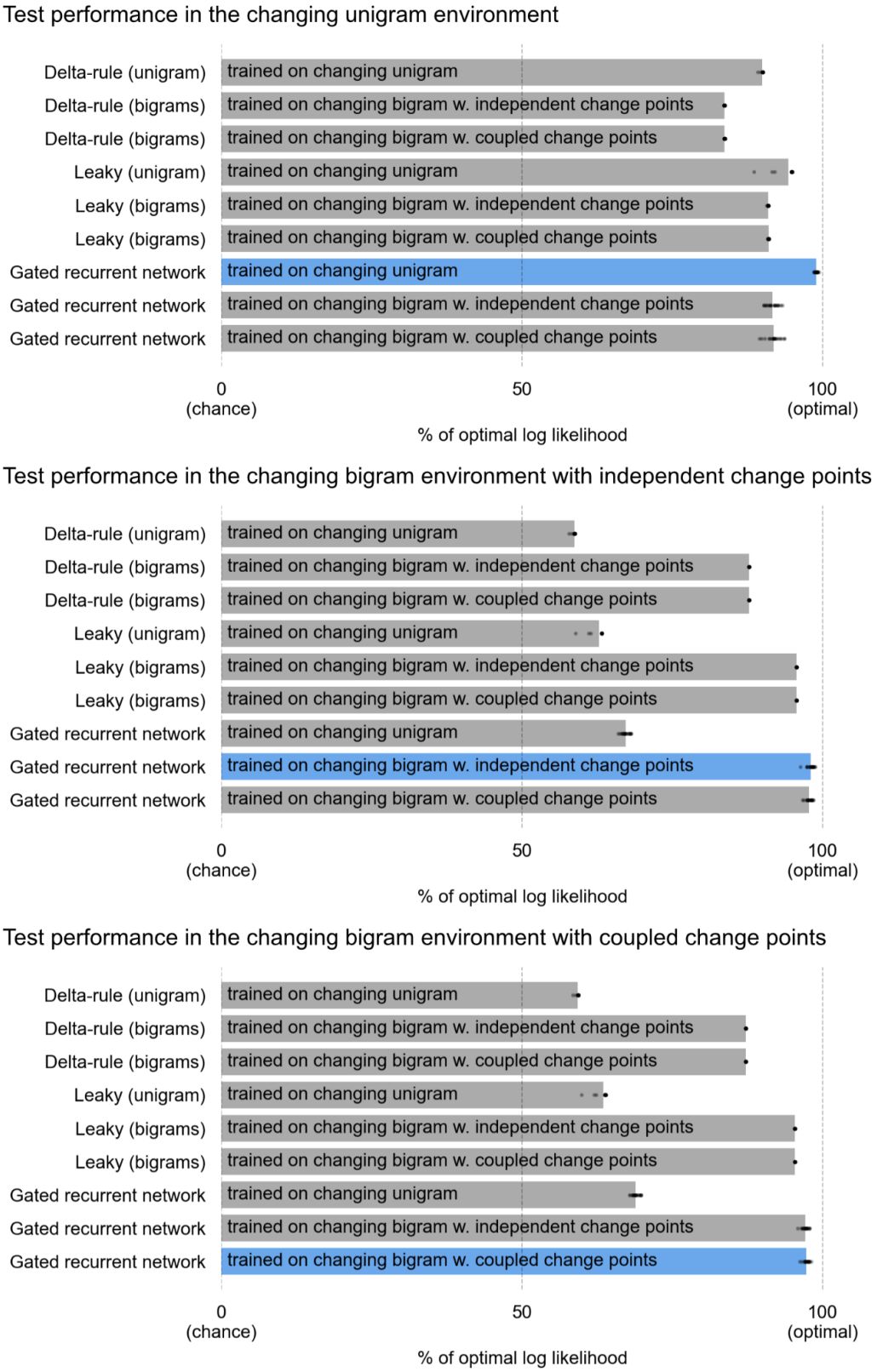
* Corresponding authors: cedric.foucault@gmail.com and florent.meyniel@cea.fr



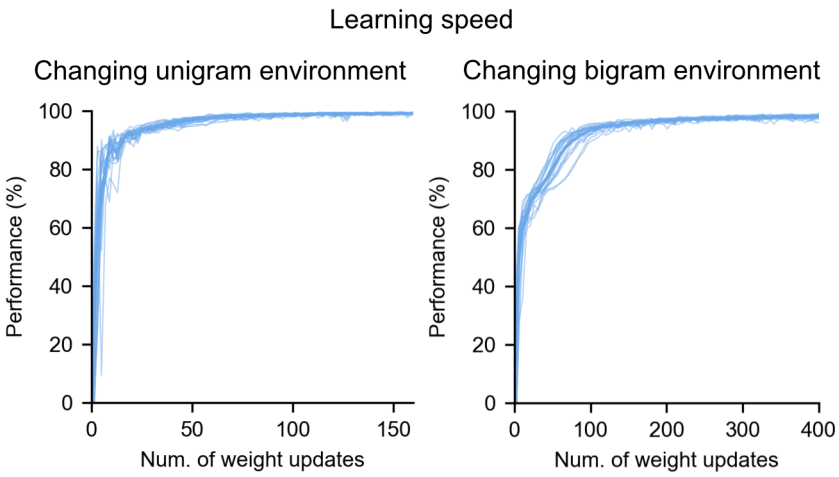
Supplementary Figure 1 | Graphical model of the generative process of each environment. Nodes encode the variables and edges the conditional dependencies between variables. Each graph represents a factorization of the joint probability distribution of all variables in the generative process: this joint distribution is the product of the conditional probability distributions of each variable given its parents in the graph. For further details on the generative processes, see Methods. In all environments, inferring the next observation from previous observations using such a graph is computationally difficult because it requires computing and marginalizing over the continuous probability distribution of the latent probabilities. This distribution is not easy to compute because it incorporates the likelihoods of the observations (for any latent probability value) and the change point probabilities from all previous time steps, and requires normalization. Notice also the increasingly complex conditional structures of the graphs from left to right. In the unigram environment, observations are conditionally independent given the latent probabilities, but in the bigram environments, they interact. In the bigram environment with coupled change points, the hierarchical structure implies that the two latent bigram probabilities are no longer conditionally independent of each other given their values at the previous time step, since they are connected by a common parent (the change point).



Supplementary Figure 2 | Attunement of the effective learning rate to the change point probabilities. **a**, Average effective learning rate of the gated recurrent networks as a function of the change point probability used during testing (columns) and during training (rows). Each row corresponds to a different set of 20 networks trained in the changing unigram environment with the indicated change point probability. Each column corresponds to a different test set with the indicated change point probability, each of 1,000 out-of-sample sequences. The networks' effective learning rate was measured and averaged over time, sequences, and networks. **b,** Average effective learning rate of the optimal agent as a function of the change point probability used during testing (columns) and the prior on the change point probability assumed by the model (rows). The optimal agent was tested on the same sets of sequences as the gated recurrent networks and its effective learning rate was averaged over time and sequences.



Supplementary Figure 3 | Performance across training and test environments. For each type of agent and each environment, a set of 20 agents was trained in the given environment as in Fig. 2, 5, and 6. The performance of each set of trained agents was then evaluated in each test environment, using 1,000 new sequences per environment and the same performance measure as in Fig. 2 and 5.



Supplementary Figure 4 | Learning speed of the gated recurrent networks in the changing unigram and bigram environments. During training, the networks' weights were iteratively updated, with each update based on the evaluation of the cost function on 20 sequences. Prediction performance was repeatedly measured after each iteration as the % of optimal log likelihood on an out-of-sample validation set of 200 sequences. The thin lines and the thick line show the mean and the individual performances of the 20 networks, respectively.

Supplementary Table 1 | Selected hyperparameter values after optimization. (*: fixed value.)

environment	network type	N	η_0	$\sigma_{x.}$	$\sigma_{h.}$	$\mu_{h.,ii}$
unigram	gated recurrent network	3	8.00E-02	0.02	0.02	0*
unigram	gated recurrent network	11	6.60E-02	0.43	0.21	0*
unigram	gated recurrent network	45	4.20E-02	1	0.02	0*
unigram	without gating	3	2.50E-02	1	0.07	0*
unigram	without gating	11	1.70E-02	1	0.07	0*
unigram	without gating	45	7.60E-03	1	0.08	0*
unigram	without lateral connections	3	5.30E-02	0.02	0.02	1
unigram	without lateral connections	11	2.70E-02	1	0.02	1
unigram	without lateral connections	45	1.30E-02	1	1	1
unigram	without recurrent weight tuning	3	1.00E-01	1.07	0.55	0*
unigram	without recurrent weight tuning	11	1.00E-01	2	0.41	0*
unigram	without recurrent weight tuning	45	1.00E-01	2	0.26	0*
unigram	without recurrent weight tuning	474	9.60E-03	1	0.1	0*
bigram	gated recurrent network	3	6.30E-02	0.02	1	0*
bigram	gated recurrent network	11	4.40E-02	1	0.02	0*
bigram	gated recurrent network	45	1.60E-02	1	0.02	0*
bigram	without gating	3	5.50E-02	0.02	0.13	0*
bigram	without gating	11	3.20E-02	1	0.05	0*
bigram	without gating	45	8.90E-03	1	0.06	0*
bigram	without lateral connections	3	4.30E-02	1	0.02	0
bigram	without lateral connections	11	4.30E-02	1	1	0
bigram	without lateral connections	45	2.80E-02	1	1	0
bigram	without recurrent weight tuning	3	6.60E-02	0.73	0.55	0*
bigram	without recurrent weight tuning	11	1.00E-01	2	0.45	0*
bigram	without recurrent weight tuning	45	1.00E-01	2	0.31	0*
bigram	without recurrent weight tuning	474	1.10E-02	2	0.1	0*