nature neuroscience

Article

https://doi.org/10.1038/s41593-024-01671-x

Reward prediction error neurons implement an efficient code for reward

Received: 3 November 2022

Accepted: 29 April 2024

Heiko H. Schütt[1,2,4] ✉, Dongjae Kim[1,3,4] & Wei Ji Ma[1]

We use efficient coding principles borrowed from sensory neuroscience to derive the optimal neural population to encode a reward distribution. We show that the responses of dopaminergic reward prediction error neurons in mouse and macaque are similar to those of the efficient code in the following ways: the neurons have a broad distribution of midpoints covering the reward distribution; neurons with higher thresholds have higher gains, more convex tuning functions and lower slopes; and their slope is higher when the reward distribution is narrower. Furthermore, we derive learning rules that converge to the efficient code. The learning rule for the position of the neuron on the reward axis closely resembles distributional reinforcement learning. Thus, reward prediction error neuron responses may be optimized to broadcast an efficient reward signal, forming a connection between efficient coding and reinforcement learning, two of the most successful theories in computational neuroscience.

Processing rewards is critical for much of cognition, including decision-making, planning and learning. An important reward representation in the brain is maintained by reward prediction error neurons (RPENs)[1]. These dopaminergic neurons in the midbrain respond to received rewards relative to an expectation based on past experience. The existence of these neurons is the most prominent evidence in favor of reinforcement learning (RL) in the brain[2]. Moreover, RPENs have been implied in a broad range of tasks that require value-based cognition[3]. Thus, the encoding of reward by RPENs is a cornerstone of our understanding of reward signals in the brain.

In a different domain of neuroscience (sensory processing), it has been hypothesized that neuronal codes are optimized for efficiency, that is, to convey as much information as possible with given budgets for the number of neurons and the number of action potentials[4,5]. The efficient coding hypothesis has long been used to account for the response properties of sensory neurons[6–8]. Gradually, the notion of efficient coding has also made inroads in the domain of reward. Efficient coding has been invoked to explain reward-based choices[9–11], and contextual modulation of the responses of cortical reward neurons has been interpreted as a form of efficient coding[12]. In a time interval estimation task, an RL agent that encoded the duration of a task-irrelevant interval with much lower resolution than that of a task-relevant interval could account for RPEN responses and their relation to behavior[13].

Here, we investigate at the neuronal level whether RPENs implement an efficient code for reward value. We first derive the most efficient population of sigmoidally tuned neurons to encode rewards sampled from an arbitrary given distribution. We then apply this general framework to two datasets, one in mice with a fixed reward distribution[14–16] and one in monkeys with a variable reward distribution[17]. We find that key properties of the efficient code are reflected in the data, suggesting that efficient coding could serve as a unifying principle. Finally, we develop learning rules for the efficient code.

Results

We analytically derive the optimal population of neurons to encode rewards by extending the framework of Ganguli and Simoncelli[18]. Assuming that RPENs have a sigmoidal tuning curve for reward (Fig. 1a)[19,20], we start with a large family of populations, within which we search for the most efficient one. We construct the family based on a base population of neurons on the unit interval. We then allow any smooth monotone function to map from reward R to the unit interval and define the responses of the neurons in terms of the responses of the base population at the mapped location. Additionally, we allow an arbitrary scaling of neuron response gains and of the density of neurons depending on their placement. In mathematical terms, we

---

[1] Center for Neural Science and Department of Psychology, New York University, New York, NY, USA. [2] Department of Behavioural and Cognitive Sciences, Université du Luxembourg, Esch-Belval, Luxembourg. [3] Department of AI-Based Convergence, Dankook University, Yongin, Republic of Korea. [4] These authors contributed equally: Heiko H. Schütt, Dongjae Kim. ✉e-mail: heiko.schutt@uni.lu

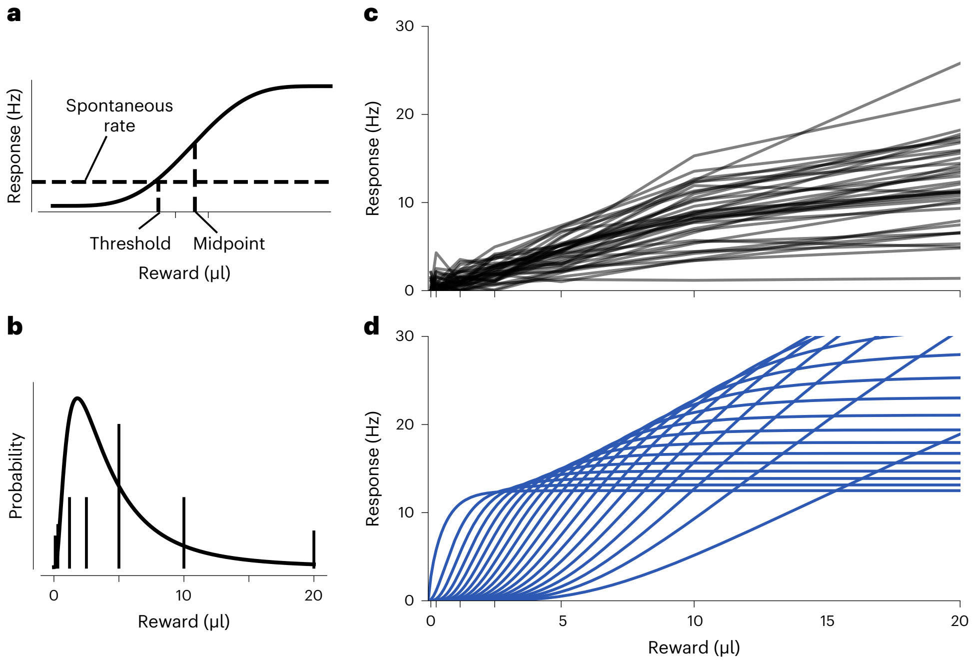


Fig. 1 | Experimental design and basic results of Eshel et al. A mouse was presented with variable rewards while RPEN responses were recorded. **a**, Terminology for sigmoidal tuning curves as expected for RPENs. The threshold is the reward value at which the spontaneous activity is reached. The midpoint is the reward value at which half the maximal response is reached. **b**, Rewards were drawn from a discrete distribution (vertical lines), which we approximate with a moment-matched log-normal distribution for our analyses (continuous line). **c**, Reward tuning curves of the 39 dopaminergic RPENs measured by Eshel et al.[15] after preprocessing of Dabney et al.[16]. We additionally subtracted the minimal response across all trials for each neuron. **d**, Efficient code for the reward distribution in **a**. For visual clarity, only 20 neurons are shown.

use a prototypical sigmoidal response function h_0 on the unit interval to define the response of a neuron with midpoint at μ as

$$h_\mu(R) = h_0[F(R) - F(\mu)]g(\mu), \tag{1}$$

where $F(R) = \int_{-\infty}^{R} f(t)dt$ is the mapping from reward space to the unit interval. Its derivative $f(R) \geq 0$ represents the conversion between reward space and the unit interval. $g(\mu)$ is the gain of a neuron with midpoint at μ. Finally, we define the density function $d(R)$ as the probability that a neuron's midpoint is placed at reward value R.

We then optimize the three functions to find the most efficient populations that can be generated this way to encode rewards from a distribution with density $p(R)$ and cumulative density $P(R)$ (Methods). Efficiency here refers to the maximization of a measure of information given constraints on the expected total firing rate. This yields a family of equally efficient populations, which we parameterized by $\alpha \in [0, 1]$:

$$f(R) = p(R); \qquad d(R) \propto \frac{p(R)}{[1 - P(R)]^{1-\alpha}}; \qquad g(R) \propto \frac{1}{[1 - P(R)]^{\alpha}}. \tag{2}$$

This solution is indeed efficient (Extended Data Fig. 1) and extends the earlier result from Ganguli and Simoncelli[18], which assumed a uniform density of neurons on the unit interval, effectively setting $d(R) = f(R)$ in our formulation. Their solution is a special case of our family, with $\alpha = 1$ and thus $d(R) = p(R)$ and $g(R) = \frac{1}{1-P(R)}$.

The efficient code captures the two main response properties of RPENs to given rewards: (1) the response increases as reward magnitude increases, and (2) the response decreases as the expected reward increases (see Extended Data Fig. 2 for effects of the reward distribution and α on the population). The former is readily apparent through Eq. (1). The latter is due to the optimization of midpoints in the efficient coding framework. When a neuron's midpoint increases, its response to all rewards decreases. To maximize information, neurons need to move their midpoints to the quantiles of the distribution. If a predictable reward is given repeatedly, the midpoints of all neurons will converge to the given reward level, and all neurons will encode the error in reward prediction relative to the midpoint, with the midpoint and threshold moving closer together. For distributions of rewards, higher rewards will still always lead to higher responses, and shifts of the whole distribution to higher values will reduce the responses of all neurons.

Application 1: variable-reward task

We first compare the predictions of the efficient coding framework to data from the 'variable-reward task' of Eshel et al.[14,15] (Fig. 1b). In this task, mice were given one of seven reward magnitudes in each trial with certain probabilities while RPEN responses were recorded. The observed RPEN reward tuning curves are shown in Fig. 1c. We derived the efficient code for a continuous approximation of the discrete probability distribution in the form of a log-normal distribution matching the mean and variance of the discrete distribution ($\mu = 1.29$ and $\sigma^2 = 0.71$). The predictions for neural density and gain are shown in Extended Data Fig. 3. To compare our predictions to the data, we use the same number of neurons as in the dataset ($n = 39$) and adjust free parameters of the derived efficient population to match the measured population. We first set α to maximize the probability of the observed midpoints (whose distribution depends only on this parameter), resulting in $\alpha = 0.673$. We define a neuron's threshold as the point where the spontaneous firing rate is surpassed.

We fitted the spontaneous firing rate r^* and k, a parameter controlling the slopes of all neurons, to the observed thresholds, resulting in the estimates $\hat{r}^* = 10.08$ and $\hat{k} = 5.20$. Finally, we set the constraint on the expected spike rate, $r_{\max}$, to match the average neural gain based on sigmoid fits to the tuning curves, resulting in $\hat{r}_{\max} = 322.5$ (see Methods for details). A subset of the resulting efficient code is shown in Fig. 1d. We now examine several properties of the efficient code and compare them to the data.

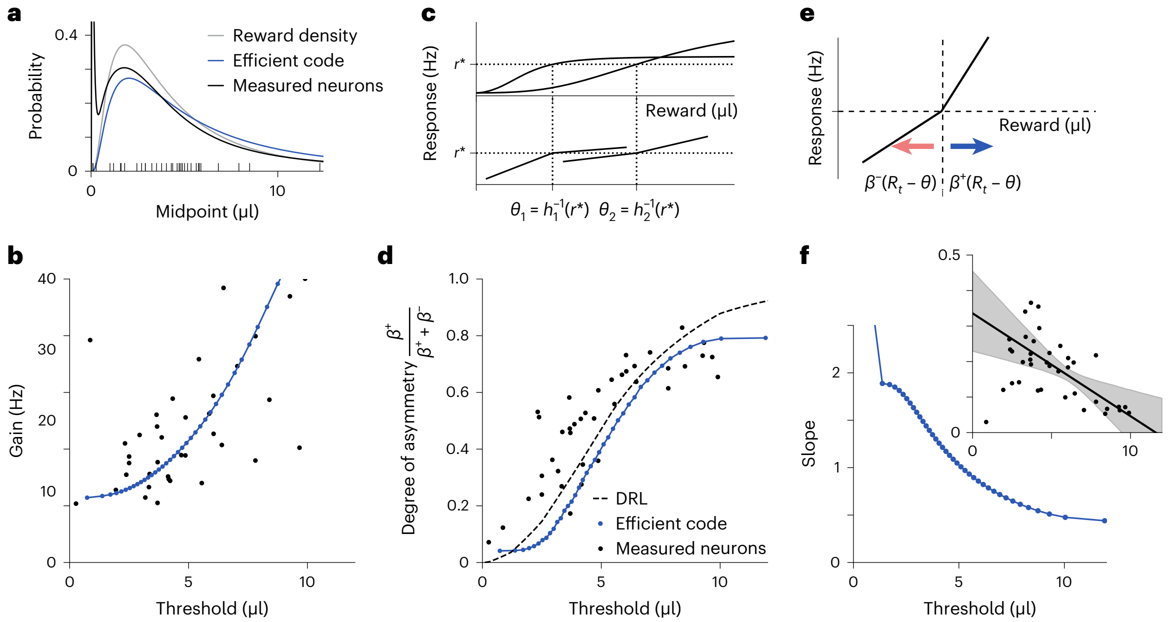


Fig. 2 | Comparisons between the measured neurons of Eshel et al. and the efficient code. a, The distribution of the midpoints of the measured neurons covers the reward distribution with a slight upward shift (mean midpoint = 5.96 versus mean reward = 5.21; $t_{38} = 0.848$, $P = 0.402$, two-sided t-test, no multiple comparisons); the efficient code accounts for this (mean = 5.36). **b**, RPEN gain, that is, maximal response of a fitted sigmoid function, plotted against threshold. There is a significant positive relationship ($r_{37} = 0.637$, $P = 1.277 \times 10^{-5}$, two-sided test, no multiple comparisons, normality assumption not tested). The neurons of an efficient population (blue) again match the measured population (black) quite well. **c**, Efficient coding accounts for the relationship between asymmetry and threshold. The spontaneous activity r^* is reached at θ_1 within the concave part of the sigmoidal response function for a neuron with low threshold and gain and at θ_2 within the convex part of a sigmoidal response function for a neuron with high threshold and gain. **d**, Degree of asymmetry of neural responses ($\frac{\beta^+}{\beta^- + \beta^+}$) plotted against threshold. There is a strong positive relationship between these two variables ($r_{37} = 0.832$, $P = 5.55 \times 10^{-11}$, two-sided test, no multiple comparisons, normality assumption not tested). The neurons of an efficient population (blue) closely approximate the measured population (black). The dashed line is expected from ideal distributional temporal difference learning. **e**, DRL accounts for the relationship between asymmetry and threshold. A reward value R_t shifts the response function down proportional to β^- if below the threshold ($R_t < \theta$) and up proportional to β^+ if above the threshold ($R_t \geq \theta$). An equilibrium is achieved when the two expected shifts cancel each other. **f**, Slope of the sigmoid fit a plotted against threshold for the efficient code neurons. There is a significant negative linear relationship ($r_{37} = -0.915$, $P = 3.59 \times 10^{-16}$, two-sided test, no multiple comparisons, normality assumption not tested). The inset shows the same plot for the measured neurons. There is a significant negative linear relationship in the data ($r_{37} = -0.550$, $P = 0.000290$, two-sided test, no multiple comparisons, normality assumption not tested; the shaded region displays a 95% confidence interval for the prediction of the linear regression model).

Prediction 1: RPEN midpoints cover the reward distribution. In the efficient code, RPEN midpoints are placed at specific quantiles of the distribution with a slight bias toward higher quantiles, yielding a distribution similar to the original reward distribution. Indeed, the midpoints of the observed neurons cover the range of the reward distribution, with a roughly similar distributional shape (Fig. 2a). The mean of the measured midpoints is 5.96, higher than the mean reward, which is 5.21 ($t_{38} = 0.848$, $P = 0.40$). The fact that this difference is not significant is unsurprising; as for the measured number of neurons, even the predicted midpoints are not significantly different from the mean reward ($t_{38} = 0.246$, $P = 0.81$). The thresholds of neurons behave similar to the midpoints (Extended Data Fig. 4).

Prediction 2: RPEN gain increases with threshold. In the efficient code, the gain is higher for neurons with higher reward thresholds (Fig. 1). The intuition is that those neurons respond to fewer rewards and can thus afford a higher gain with the same expected number of spikes. This effect does not occur for unimodal tuning functions[18,21]. We observe the expected increase of gain in the data (Fig. 2b). To our knowledge, this empirical finding has not been reported before.

Prediction 3: RPEN tuning curve asymmetry flips with increasing threshold. In the efficient code, the increase of gain with threshold has implications for the shape of the RPEN tuning curves around threshold (Fig. 2c). As the gain increases, the spontaneous firing rate falls lower in the sigmoid shape. Thus, the threshold moves from the upper concave part of the sigmoid down to the convex part. In other words, neurons with low thresholds have concave tuning curves around their threshold, and neurons with high thresholds have concave ones. To quantify the degree of such 'asymmetry' of the neural responses, we followed previous work[16] and approximated the neural tuning curves with two linear response functions, one above and one below the spontaneous firing rate r^* (Fig. 2c). This allows for an asymmetric RPEN response, with the slope above the threshold β^+ being different from the slope below the threshold β^- (refs. 22,23). The ratio of slopes, $\frac{\beta^+}{\beta^- + \beta^+} \in [0,1]$, can then be used as an index of asymmetry, which is close to 1 for concave tuning curves and close to 0 for convex tuning curves. Using this index, the efficient code predicts that asymmetry is higher for neurons with higher thresholds; the shape flips from convex to concave. This relationship is also observed in the data (black dots in Fig. 2d), as first reported by Dabney et al.[16].

The observed relationship can also be accounted for by the theory of distributional RL (DRL; dashed line in Fig. 2d), which attributes it to an asymmetry of the update equations of the thresholds, such that neurons converge to different expectiles of the reward distribution (Fig. 2e). We will examine the relationship between efficient coding and DRL in detail below.

Prediction 4: RPEN slope decreases with threshold. In the efficient code, neurons with higher thresholds have a lower slope parameter of the sigmoid fit (Fig. 2f). This occurs because the predicted tuning functions are shallower in regions with lower reward density. In the

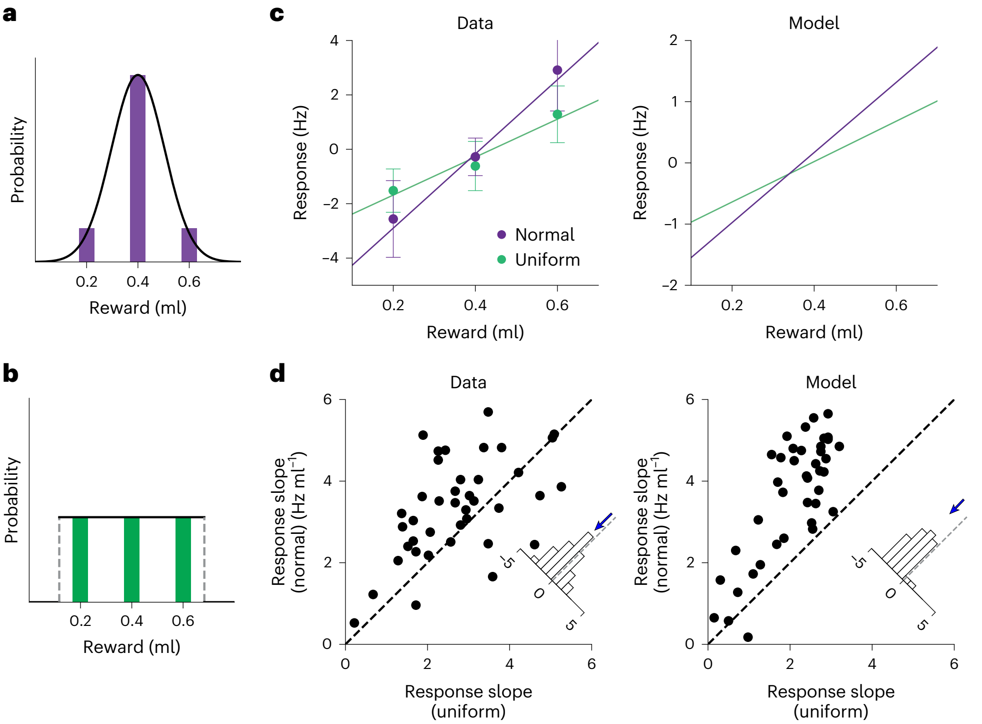


Fig. 3 | Efficient coding accounts for response slope characteristics in the variable-distribution task. a, 'Normal' distribution used in the experiment (purple) and moment-matched continuous distribution. **b**, Same for the 'uniform' distribution. Vertical dashed lines mark the boundaries of the uniform distribution. **c**, Left, example neuron from Rothenhoefer et al.[17] showing steeper tuning for the 'normal' distribution than for the 'uniform' distribution. Error bars represent ±1 s.e.m. across trials. Right, example neuron from the efficient code. **d**, Scatter plots of the response slopes of all neurons in the data ($n = 40$ (ref. 17); left) and in the efficient code (right). The dashed diagonal marks the identity line. Insets show the histograms of differences, and the downward arrows indicate the mean. The data show a significantly different response slope with a mean difference of −0.605 ($t_{38} = -3.26$, $P = 0.002$, two-sided test, no multiple comparisons, normality assumption not tested), and the model also demonstrated a significantly different response slope with a mean difference of −1.58 ($t_{38} = -10.4$, $P = 1.13 \times 10^{-12}$, two-sided test, no multiple comparisons, normality assumption not tested).

measured neurons, we indeed find a significant negative correlation of slope and threshold ($r_{37} = -0.550$, $P < 0.001$). To our knowledge, this empirical finding has also not been reported before. Overall, the measured neurons have shallower tuning curves than predicted by the efficient code. A possible explanation is that we did not take into account stochasticity in the RPEN inputs. Such stochasticity would produce random horizontal shifts of the sigmoid tuning curve, which, when averaged, would result in a lower slope.

Application 2: variable-distribution task

So far, we have examined RPEN responses to a fixed reward distribution. Rothenhoefer et al.[17] instead exposed two macaque monkeys to cues associated with rewards drawn from one of two distributions (either a 'uniform' or a 'normal' distribution) with the same mean (Fig. 3a,b). The main finding of Rothenhoefer et al. was that dopamine responses are amplified for rare rewards, producing steeper response functions for the 'normal' distribution than for the 'uniform' distribution (Fig. 3c,d). This suggests that RPENs encode the frequency of rewards.

The variable distributions make this dataset an interesting additional test for the efficient coding framework. We derived an efficient code for a moment-matched continuous approximation of each distribution (Fig. 3a,b). We fitted the parameters α, k and $r_{\max}$ to minimize the mean squared error between the model and data on the response slopes for the individual distributions. The resulting estimates were $\hat{\alpha} = 0.473$, $\hat{k} = 2.00$ and $\hat{r}_{\max} = 28.2$ (Extended Data Fig. 5). As in 'Application 1: variable-reward task', we find that the midpoints of the neurons in the efficient code cover the range of the reward distribution (Extended Data Fig. 5b,d). Moreover, the efficient coding model successfully reproduced the main finding (Fig. 3c,d), where the majority of neurons in the efficient code (39 of 40) exhibited steeper slopes for the 'normal' distribution, in agreement with the data (31 of 40).

Learning rules for the efficient code

DRL[16] proposes that RPENs update their thresholds in proportion to the difference between the received reward and their thresholds, with different proportionality factors for positive and negative differences (Fig. 2e). This learning rule leads those thresholds to converge to the expectiles of the reward distribution. This is very similar to the prediction of the efficient code, which places the midpoints of neurons at specific quantiles of the distribution. With the additional assumption that the slope of the reward tuning curve above and below threshold is proportional to the learning rate, DRL accounts for the observed relationship between threshold and asymmetry. However, without modification, it does not account for the relationship between gain and threshold or the relationship between slope and threshold. Because the efficient code does account for these observations, a good way to consolidate the two theories is to extend the DRL rule such that the gain and slope of neurons also converge to the parameters for the efficient code.

To learn the placement of neurons on the reward axis, we can use the asymmetric DRL rules. To converge to the quantiles of the distribution, we shift the neuron to higher reward values by a step size b and toward lower reward values by a different step size a if a reward below the midpoint occurs (Fig. 4a). If we set a and b such that $ap = b(1-p)$, this procedure converges to the pth quantile of the distribution. In the DRL formulation, neurons instead update their position proportional to the difference between their location and the presented reward and thus converge to the expectiles of the distribution. This changes

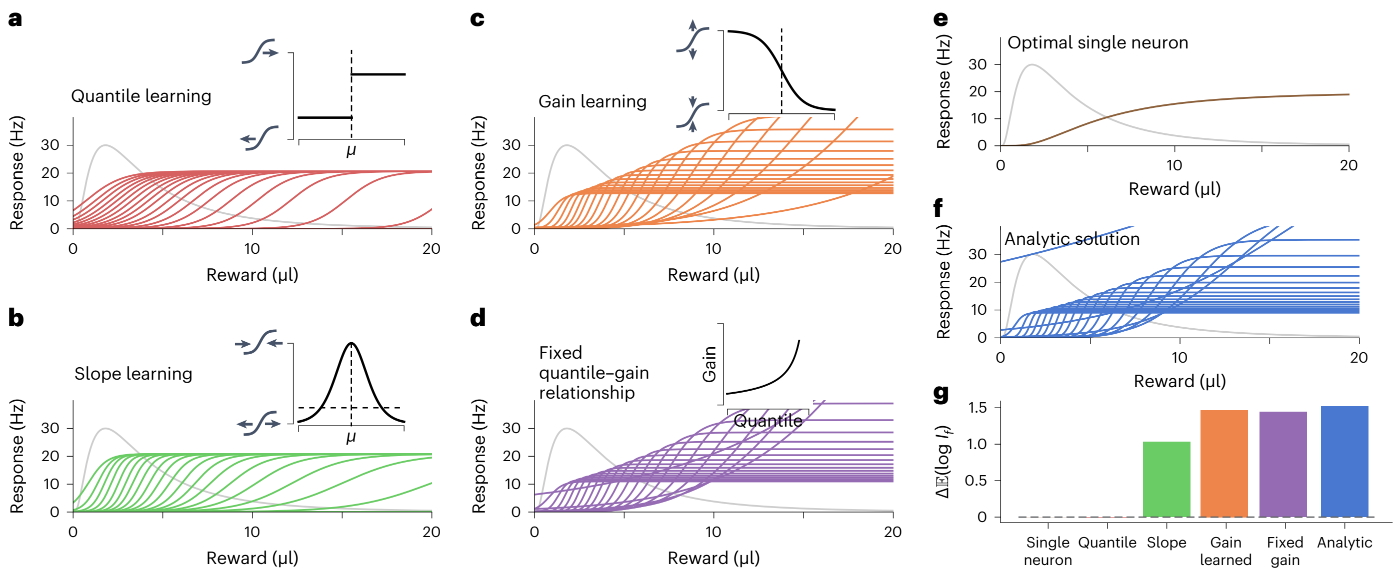


Fig. 4 | Combination of learning rules to learn the efficient code. Each graph shows the steady-state distribution population of 20 neurons after learning based on 20,000 reward presentations, with insets illustrating the learning rule. **a**, Learning the position on the reward axis for the neurons to converge to the quantiles of the distribution. This learning rule is similar to the DRL rule. **b**, Additionally learning the slope of the neurons to be proportional to the local density by increasing the slope when the reward falls within the dynamic range and decreasing otherwise. **c**, First method to set the gain: iterative adjustment to converge to a fixed average firing rate. **d**, Second method to set the gain: use a fixed gain per neuron based on the quantile it will eventually converge to. **e**, Efficient tuning curve for a single neuron. **f**, Analytically derived optimal solution. **g**, Comparison of the different populations in the overall information transfer with the same number of neurons (20) and expected firing rate (8.27 per neuron as in the fit to the measured data).

the population only slightly but achieves similar efficiency as the populations placed at the quantiles (Extended Data Fig. 6). It should also be kept in mind that both the step function that we apply for the quantiles and the linear weighting functions that yield expectiles are idealizations.

In the efficient code, the RPEN slope should be proportional to the local probability density of rewards. To learn the slope, we can estimate the local probability density based on the probability that a reward within the dynamic range of the neuron occurs, that is, in the range where the neuron has a high slope. As the dynamic range is inversely proportional to the slope, it should be inversely proportional to the local reward density as well, which means that the probability of a reward in the dynamic range should be constant. A simple learning rule to achieve this is to increase the slope and shrink the dynamic range when a reward falls into the dynamic range and decrease the slope and increase the dynamic range when a reward falls outside the dynamic range. When we allow a gradual definition for falling into the dynamic range, a natural choice is to use the derivative of the sigmoid function; this is the kernel we chose for our simulations (Fig. 4b). Any other kernel around the midpoint of the sigmoid whose width scales inversely with the slope would also work.

Finally, we need to adjust the gain of the neurons to match the efficient coding solution. For this purpose, we present two solutions that work similarly well but that are based on different biological mechanisms. In the first method, we learn an estimate of the expected firing rate and set the gain based on this estimate. Effectively, this results in the inverse of the sigmoid shape for updating the gain, with high rewards reducing a neuron's gain and low rewards increasing a neuron's gain (Fig. 4c). In the second method, we couple the fixed gain to the quantile the specific neuron should converge to based on its learning asymmetry (Fig. 4d).

These learning rules for midpoints, slope and gain together produce a population of neurons similar to the analytical efficient coding solution (Fig. 4c,d,f) with similar efficiency (Fig. 4g). These populations clearly beat the best possible solution that assumes the same tuning function for every neuron (Fig. 4e,g). Thus, these rules successfully extend the DRL rule to produce an efficient code.

Discussion

We have presented evidence that RPENs implement an efficient code for reward. Starting from a normative account of the tuning of RPEN responses, we were able to account for five empirical findings across two tasks. Thus, efficient coding could serve as a unifying principle underlying the response characteristics of RPENs.

Dependence on additional assumptions

Our results are robust to changing the measure of information used for optimizing the population, the assumed distribution of neural responses or the assumption of a sigmoid shape. Different objective functions or different response distributions lead to different relationships between the density of the reward distribution and the optimal density and gain of neurons[18]. However, neural density always increases with reward density, as it is more efficient to focus neural resources on probable rewards. Analogously, all objectives lead to higher gains at higher thresholds, as gain increases are 'cheaper' for neurons with higher thresholds. Furthermore, changing the mathematical form of the shape of the response function does not change the density and gain predictions, as those were derived independently of the shape, and only leads to slightly different predictions for the asymmetry of the responses around the threshold. For any shape that is convex in its lower response range and concave in its upper response range, the qualitative argument for the dependence between threshold and asymmetry will hold.

Limitations

Our work has limitations. To simplify our analysis, we restricted ourselves to a task with a single reward supplied in each independent episode without any action required by the animal. In this simple case, the reward received and the change in value of the current state are confounded. Thus, all our analyses are agnostic to this distinction.

In multistep[24] and delayed-reward tasks[25], RPENs take future rewards into account with temporal discounting. The efficient coding framework that we presented here would need to be modified for a temporally discounted value code. Moreover, our theory covers only the phasic reward prediction error response of RPENs. The ramping activity of RPENs (for example, refs. 26,27) will require additional explanation, most likely including an encoding of reward timing[28]. In fact, it has been proposed that RPENs encode elapsed time more generally[29] and that this encoding is tuned to task properties in a manner reminiscent of efficient coding[13].

Compatibility with RL

The efficient coding hypothesis is compatible with RL. Specifically, we extended the learning rule from DRL[16] to account not only for the relationship between asymmetry and threshold but also for the three other tuning properties predicted by the efficient code. Besides DRL, there are two alternative explanations of the data by Eshel et al.[14]: the Laplace code[30] and normalized RL[31]. In the Laplace code, temporal difference learning neurons with different parameters are used to encode the timing and a whole distribution of rewards in the future by representing an analog of the Laplace transform in the neural responses. Individual neurons still have a sigmoid response curve in this code, and the dependence between asymmetry and threshold can be created similar to as in the efficient code by cutting the curve at different response levels. In their formulation, they subtract the expected response from each neuron instead of changing the gain (as we predict and observe). Normalized RL[31] proposes that RPENs perform divisive normalization with different half saturation constants. This yields sigmoid (Naka–Rushton[32]) neurons with different thresholds, and the asymmetry around those thresholds is explained again by cutting the sigmoids at different heights. Neither of these explanations explicitly examines code efficiency or attempts to account for the dependencies of gain or slope on threshold.

Future work

Future work could examine how learning rules that produce an efficient code can be implemented in biologically realistic circuits (perhaps similar to refs. 33–36), empirically distinguish between candidate learning rules and examine how RPENs can switch from one reward distribution to another relatively quickly.

Online content

Any methods, additional references, Nature Portfolio reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41593-024-01671-x.

References

1. Schultz, W., Dayan, P. & Montague, P. R. A neural substrate of prediction and reward. *Science* **275**, 1593–1599 (1997).
2. Sutton, R. S. & Barto, A. G. *Reinforcement Learning: An Introduction* (MathWorks, 2018).
3. Balleine, B. W., Daw, N. D. & O'Doherty, J. P. in *Neuroeconomics* (eds Glimcher, P. W. et al.) 367–387 (Academic Press, 2009).
4. Attneave, F. Some informational aspects of visual perception. *Psychol. Rev.* **61**, 183–193 (1954).
5. Barlow, H. B. in *Sensory Communication* (ed Rosenblith, W. A.) 216–234 (MIT Press, 1961).
6. Laughlin, S. A simple coding procedure enhances a neuron's information capacity. *Z. Naturforsch. C Biosci.* **36**, 910–912 (1981).
7. Schwartz, O. & Simoncelli, E. P. Natural signal statistics and sensory gain control. *Nat. Neurosci.* **4**, 819–825 (2001).
8. Wei, X.-X. & Stocker, A. A. Lawful relation between perceptual bias and discriminability. *Proc. Natl Acad. Sci. USA* **114**, 10244–10249 (2017).
9. Louie, K., Glimcher, P. W. & Webb, R. Adaptive neural coding: from biological to behavioral decision-making. *Curr. Opin. Behav. Sci.* **5**, 91–99 (2015).
10. Polanía, R., Woodford, M. & Ruff, C. C. Efficient coding of subjective value. *Nat. Neurosci.* **22**, 134–142 (2019).
11. Bhui, R., Lai, L. & Gershman, S. J. Resource-rational decision making. *Curr. Opin. Behav. Sci.* **41**, 15–21 (2021).
12. Louie, K. & Glimcher, P. W. Efficient coding and the neural representation of value. *Ann. N Y Acad. Sci.* **1251**, 13–32 (2012).
13. Motiwala, A., Soares, S., Atallah, B. V., Paton, J. J. & Machens, C. K. Efficient coding of cognitive variables underlies dopamine response and choice behavior. *Nat. Neurosci.* **25**, 738–748 (2022).
14. Eshel, N. et al. Arithmetic and local circuitry underlying dopamine prediction errors. *Nature* **525**, 243–246 (2015).
15. Eshel, N., Tian, J., Bukwich, M. & Uchida, N. Dopamine neurons share common response function for reward prediction error. *Nat. Neurosci.* **19**, 479–486 (2016).
16. Dabney, W. et al. A distributional code for value in dopamine-based reinforcement learning. *Nature* **577**, 671–675 (2020).
17. Rothenhoefer, K. M., Hong, T., Alikaya, A. & Stauffer, W. R. Rare rewards amplify dopamine responses. *Nat. Neurosci.* **24**, 465–469 (2021).
18. Ganguli, D. & Simoncelli, E. P. Efficient sensory encoding and Bayesian inference with heterogeneous neural populations. *Neural Comput.* **26**, 2103–2134 (2014).
19. Fiorillo, C. D., Tobler, P. N. & Schultz, W. Discrete coding of reward probability and uncertainty by dopamine neurons. *Science* **299**, 1898–1902 (2003).
20. Cohen, J. D. & Servan-Schreiber, D. A theory of dopamine function and its role in cognitive deficits in schizophrenia. *Schizophr. Bull.* **19**, 85–104 (1993).
21. Wei, X.-X. & Stocker, A. A. Bayesian inference with efficient neural population codes. In *Artificial Neural Networks and Machine Learning—ICANN 2012*, Vol. 7552 (eds Hutchison, D. et al.) 523–530 (Springer, 2012).
22. Frank, M. J., Seeberger, L. C. & O'Reilly, R. C. By carrot or by stick: cognitive reinforcement learning in Parkinsonism. *Science* **306**, 1940–1943 (2004).
23. Mikhael, J. G. & Bogacz, R. Learning reward uncertainty in the basal ganglia. *PLoS Comput. Biol.* **12**, e1005062 (2016).
24. Kobayashi, S. & Schultz, W. Influence of reward delays on responses of dopamine neurons. *J. Neurosci.* **28**, 7837–7846 (2008).
25. Roesch, M. R., Calu, D. J. & Schoenbaum, G. Dopamine neurons encode the better option in rats deciding between differently delayed or sized rewards. *Nat. Neurosci.* **10**, 1615–1624 (2007).
26. Kim, H. R. et al. A unified framework for dopamine signals across timescales. *Cell* **183**, 1600–1616 (2020).
27. Starkweather, C. K. & Uchida, N. Dopamine signals as temporal difference errors: recent advances. *Curr. Opin. Neurobiol.* **67**, 95–105 (2021).
28. Starkweather, C. K., Babayan, B. M., Uchida, N. & Gershman, S. J. Dopamine reward prediction errors reflect hidden-state inference across time. *Nat. Neurosci.* **20**, 581–589 (2017).
29. Soares, S., Atallah, B. V. & Paton, J. J. Midbrain dopamine neurons control judgment of time. *Science* **354**, 1273–1277 (2016).
30. Tano, P., Dayan, P. & Pouget, A. A local temporal difference code for distributional reinforcement learning. In *Advances in Neural Information Processing Systems 33* (eds Larochelle, H. et al.) 13662–13673 (Neural Information Processing Systems Foundation, 2020).
31. Louie, K. Asymmetric and adaptive reward coding via normalized reinforcement learning. *PLoS Comput. Biol.* **18**, e1010350 (2022).

32. Naka, K. I. & Rushton, W. A. H. An attempt to analyse colour reception by electrophysiology. *J. Physiol.* **185**, 556–586 (1966).
33. Bredenberg, C., Simoncelli, E. P. & Savin, C. Learning efficient task-dependent representations with synaptic plasticity. In *Advances in Neural Information Processing Systems 33* (eds Larochelle, H. et al.) 15714–15724 (Neural Information Processing Systems Foundation, 2020).
34. Savin, C. & Triesch, J. Emergence of task-dependent representations in working memory circuits. *Front. Comput. Neurosci.* **8**, 57 (2014).
35. Gerstner, W., Lehmann, M., Liakoni, V., Corneil, D. & Brea, J. Eligibility traces and plasticity on behavioral time scales: experimental support of neoHebbian three-factor learning rules. *Front. Neural Circuits* **12**, 53 (2018).
36. Frémaux, N. & Gerstner, W. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Front. Neural Circuits* **9**, 85 (2016).

Methods

Analysis of neural data

Variable-reward task. We used the data from the variable-reward task by Eshel et al.[14] as preprocessed by Dabney et al.[16]. For details of data acquisition, please refer to Eshel et al.[14]. Five mice were subjected to a variable-reward paradigm incorporating two experimental conditions: a no-odor condition and an odor condition in which an odor was presented to cue the reward delivery. During each trial, one of seven reward magnitudes (0.1, 0.3, 1.2, 2.5, 5, 10 and 20 μl) was randomly chosen with probabilities of 0.066, 0.091, 0.15, 0.15, 0.31, 0.15 and 0.077, respectively. Electrophysiological data were recorded from $n = 40$ ventral tegmental area dopaminergic RPENs. From each of the mice, 3, 6, 9, 16 and 6 neurons were selected, respectively. One neuron was excluded because it never showed a response larger than its baseline for any reward magnitude. We used the data as preprocessed by Dabney et al.[16]. As part of this preprocessing, on each trial, a prestimulus response was subtracted from the later responses, and an offset was added to the neural responses in the no-odor condition to equalize the two conditions. As this procedure yielded negative firing rates, we additionally subtracted for each neuron the overall smallest response from all its responses.

Variable-distribution task. To further investigate the responses of dopaminergic RPENs with different reward distributions, we analyzed the data from the variable-distribution task presented in Rothenhoefer et al.[17]. For detailed data acquisition methods, we refer to that paper. Two macaque monkeys were subjected to a variable-reward paradigm, where cues indicated rewards from either a uniform or a normal distribution. In each trial, rewards of 0.2, 0.4 or 0.6 ml were delivered based on a uniform distribution (1/3 chance for each amount) or a normal distribution where rewards of 0.2 and 0.6 ml had a 2/15 probability and a reward of 0.4 ml had an 11/15 probability. Electrophysiological data from RPENs were collected to monitor dopamine responses. The peristimulus response was subtracted from the later responses and smoothed as in the variable-reward task.

Logistic fits. For each neuron separately, we fitted logistic sigmoid functions $\sigma(R) = \frac{a_1}{1-e^{-a_2(x-a_3)}}$ to the responses. We denote the gain a_1, the slope a_2 and the midpoint a_3 collectively by $\mathbf{a}$. To estimate the parameters, we used maximum-likelihood estimation based on Poisson noise, so that

$$\hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmin}} \sum_{j=1}^{T} \left(r_j \log\left(\sigma(R_j; \mathbf{a})\right) - \sigma(R_j; \mathbf{a})\right), \tag{3}$$

where r_j is the measured firing rate of a neuron responding to the reward magnitude R_j on the jth trial. We proceeded to compare the fitted parameters between the data and the efficient code.

Fitting asymmetric slopes. To enable comparisons to the asymmetry data from Dabney et al.[16], we repeated their analysis and applied the same asymmetry calculation to the simulated neurons in our efficient code.

First, we subtracted a baseline from the smoothed peristimulus time histograms for each neuron and trial. The baseline was the mean firing rate over the 1,000 ms before stimulus onset. We then estimated asymmetric slopes, β^+ and β^-, by separately fitting linear functions to the responses above and below threshold. Following Dabney et al.[16], we used estimated utility space as the x axis for this step, but the results are not sensitive to the choice of utility function.

In math terms, the slope above threshold β^+ and the slope below threshold β^- for responses y_j to rewards R_j are thus defined as

$$\beta_i^+ = \underset{\beta_i}{\operatorname{argmin}} \sum_{j=1}^{T} \left[r^* + \beta_i\left(R_j - \theta\right) - y_j\right]^2, \text{ for } R_j \geq \theta$$

$$\beta_i^- = \underset{\beta_i}{\operatorname{argmin}} \sum_{j=1}^{T} \left[r^* + \beta_i\left(R_j - \theta\right) - y_j\right]^2, \text{ for } R_j \leq \theta,$$

where r^* is the spontaneous firing rate and the sum over j runs over the T trials.

To achieve more efficient fitting than Dabney et al.[16], we derived analytical solutions for β^+ and β^- for a given θ, which are

$$\beta_i^+ = \frac{\sum_{j=1}^{T}(R_j - \theta)(y_j - r^*)}{\sum_{j=1}^{T}(R_j - \theta)^2} \text{ for } R_j \geq \theta \tag{4}$$

$$\beta_i^- = \frac{\sum_{j=1}^{T}(R_j - \theta)(y_j - r^*)}{\sum_{j=1}^{T}(R_j - \theta)^2} \text{ for } R_j \leq \theta. \tag{5}$$

Thresholds (called reversal points in Dabney et al.[16]) were then chosen to minimize the overall sum of squares by sampling 20,000 thresholds from a uniform distribution between the range of rewards and choosing the best-fitting one. Dabney et al. used the same random sampling procedure for β^+ and β^-.

For fitting our simulated neurons, we set the threshold $\theta = h_i^{-1}(r^*)$ and $y_j = h_i(v_j)$ and fit the β values based on the results for 10^5 trials each above and below threshold sampled from the approximated log-normal reward distribution restricted to the range above and below threshold, respectively.

Statistics and reproducibility

We used standard statistical tests based on normal distribution assumptions throughout, even though our hypothesis is much more detailed than predicting the existence of a difference. For the tests, the data distribution was assumed to be normal, but this was not formally tested.

All data we use here were already publicly available. Thus, no statistical method was used by us to predetermine sample size. We followed the original authors in excluding one neuron that never responded above its spontaneous rate. The experiments were not randomized, as there were no experimental groups to assign to randomly. The investigators who collected the data were blind to our hypothesis as this hypothesis was not stated when the data were recorded, but they were not blind to any aspect of the experimental design. Data collection and analysis were not performed blind to the conditions of the experiments.

Derivation of the efficient code

To derive the efficient code for reward, we extended the framework for analytical solutions proposed by Ganguli and Simoncelli[18]. In this framework, a population is uniformly defined not on a specific numerical interval, as was done by Ganguli and Simoncelli (who used the interval $[0, N]$, with N being the number of neurons, which is entirely equivalent), but on the unit interval $[0, 1]$. By using an adaptable mapping between the unit interval and the stimulus space and functions that transform the population, we can define a broad class of populations among which we can find the most efficient ones analytically. We extended this framework by allowing the distribution of neurons to be nonuniform, which decouples the local slope and density of neurons that are coupled in the original framework.

Definitions. We start by defining a ‘standard’ population of RPENs. This standard population is a collection of sigmoidal tuning curves s_ω, $[0, 1] \rightarrow [0, 1]$, indexed by their ‘center’ $\omega \in [0, 1]$. Individual neurons are assumed to be monotonically increasing functions whose derivatives are unimodal functions with their peak near ω. Additionally, we assume that the overall Fisher information $I_F(x)$ of this population about the position $x \in [0, 1]$ and the overall increase in firing rate are independent of x, that is,

$$\int s'_\omega(x) d\omega \approx \text{const} \tag{6}$$

$$I_F(x) = \int I_F(x, \omega) d\omega \approx \text{const}, \tag{7}$$

where $I_F(x, \omega)$ is the Fisher information provided by the tuning function with center ω, and $s'_\omega(x)$ is the derivative of the tuning function.

Finally, we assume that the total amount of Fisher information provided by each neuron is approximately constant,

$$\int I_F(x, \omega) dx \approx \text{const}. \tag{8}$$

One simple way to construct a population that fulfills these criteria is to start with a prototypical sigmoid function s_0 centered on 0 and define the other tuning curves as $s_\omega(x) = s_0(x - \omega)$. This construction often leads to strong edge effects at 0 and 1 because a substantial part of the sigmoid response curves near the borders will fall outside the [0, 1] interval. We ameliorate these effects by using a more sophisticated population based on cumulative beta distributions below. Nonetheless, this simple construction illustrates the intuition behind a uniform population of sigmoidal neurons well and was actually used by Ganguli and Simoncelli in their implementation.

To map between the unit interval and stimulus space, we use a strictly monotonic function $F(R) = \int_{-\infty}^{R} f(s) ds$ from the stimulus space into the unit interval parameterized by its derivative $f(s) > 0$, which we will optimize below. As F is strictly monotonic, it is also invertible, with its inverse F^{-1} mapping the unit interval into stimulus space. Additionally, we allow a gain function $g > 0$ of stimulus space, which scales the tuning curve of neurons placed at different positions in stimulus space.

With these definitions, we can now define the tuning function h_μ of a neuron with a midpoint at stimulus level μ,

$$h_\mu(R) = g(\mu) s_{F(\mu)}(F(R)), \tag{9}$$

that is, we map both the positions of the neurons and the stimuli into the unit interval, evaluate the sigmoids there and scale the result with the gain.

Up to this point, our definitions are equivalent to those of Ganguli and Simoncelli[18], who now assume that the neurons are equally spaced in the interval. This additional assumption makes it impossible to place more neurons in some stimulus range without changing their shape. This removes one of the possible adaptations of neural populations mentioned by other authors such as, for example, Wei and Stocker[37]. To allow different distributions of the neurons, we instead assume that the μ_i are drawn from a distribution over reward space with density d, which we will optimize below. The equal placement assumed by Ganguli and Simoncelli is equivalent to the assumption $d(R) = f(R)$, which yields a uniform distribution over the placements in the unit interval ω.

Optimization objective. We further assume that the RPEN firing rates are subject to independent Poisson noise, which is the simplest noise assumption. We can then calculate the Fisher information $I_F(R; \mu)$ provided by a neuron centered on μ,

$$I_F(R; \mu) = g(\mu) \frac{s'_{F(\mu)}(F(R))^2}{s_{F(\mu)}(F(R))} f^2(R). \tag{10}$$

We then optimize the expected value (over both R and μ) of the logarithm of the Fisher information, I_F, based on N neurons,

$$I_N = \mathbb{E}_R \mathbb{E}_{\boldsymbol{\mu}} \left[\log \left(\frac{1}{N} \sum_i I_F(R; \mu_i) \right) \right] \tag{11}$$

This objective was originally proposed as a lower bound on the mutual information between the neural responses and the stimulus[18,38,39], which becomes tight for small Gaussian-shaped errors on the encoded value. However, some later publications cast doubt on this interpretation. In particular, Wei and Stocker[40] interpret this equation as an upper bound on mutual information instead. Additionally, low firing rates and/or steep changes in firing rate can lead to this approximation being highly inaccurate, and this realm is actually reached for cases in which neurons are measured only for short times[41]. Nonetheless, I_N remains the best Fisher information-based approximation to mutual information, and we follow Ganguli and Simoncelli[18] in optimizing it. They also tried a few other functionals of Fisher information and got qualitatively similar results. Mutual information itself is unfortunately untractable for these derivations.

Assuming further that there is a large population of RPENs, we can now look at this distribution in the limit of infinitely many neurons. According to the strong law of large numbers, the logarithm of the mean in Eq. (11) almost surely converges to the logarithm of the expected Fisher information, because the Fisher information provided by each single neuron has finite variance:

$$\lim_{N\to\infty} \log \left(\frac{1}{N} \sum_i I_F(R; \mu_i) \right) = \log \mathbb{E}_\mu [I_F(R; \mu)] \tag{12}$$

Because this is a constant with respect to μ, we can drop the outer expected value, yielding

$$I \equiv \lim_{N\to\infty} I_N = \mathbb{E}_R \left[\log \left(\mathbb{E}_\mu \left[I_F(R; \mu_i) \right] \right) \right]. \tag{13}$$

To make this expression amenable to optimization, we simplify it as follows:

$$\mathbb{E}_R \left[\log \left(\mathbb{E}_\mu \left[I_F(R; \mu) \right] \right) \right] \tag{14}$$

$$= \int_{-\infty}^{\infty} p(R) \log \left(\int_{-\infty}^{\infty} d(\mu) g(\mu) \frac{s'_{F(\mu)}(F(R))^2}{s_{F(\mu)}(F(R))} f^2(R) d\mu \right) dR \tag{15}$$

$$\approx \int_{-\infty}^{\infty} p(R) \log \left(d(R) g(R) f(R) \int_{-\infty}^{\infty} f(\mu) \frac{s'_{F(\mu)}(F(R))^2}{s_{F(\mu)}(F(R))} d\mu \right) dR \tag{16}$$

$$\approx \text{constant} + \int_{-\infty}^{\infty} p(R) \log \left(d(R) g(R) f(R) \right) dR. \tag{17}$$

Here, we eliminated the inner integral through two steps, which are only approximately correct. First, we observe for sigmoidal functions that $\frac{h'_0(F(R)-F(\mu))^2}{h_0(F(R)-F(\mu))}$ is a unimodal function that is centered around R, which effectively acts as a smoothing kernel that is convolved with the rest of the integral. If $d(\mu)$, $f(\mu)$ and $g(\mu)$ change little over the range of this smoothing kernel, we can replace them by their values at R and pull them out of the integral. Second, the integral over μ we are left with is (approximately) shift invariant, that is, the same for all R and invariant to the monotonic transformation F, if the original population is (approximately) uniform as defined in Eqs. (7) and (8).

The final optimization problem is now to optimize this approximation of I with respect to the three functions d, g and f:

$$I \approx \int_{-\infty}^{\infty} p(R) \log \left(d(R) g(R) f(R) \right) dR. \tag{18}$$

Additionally, we enforce that the expected number of spikes $\mathbb{E}[r]$ under the prior distribution is smaller or equal to a bound $r_{\max}$. The expected number of spikes is

$$\mathbb{E}[r] = N \mathbb{E}_R \left[\mathbb{E}_\mu [h_\mu(R)] \right] \tag{19}$$

$$= N\int_{-\infty}^{\infty} p(R)\int_{-\infty}^{\infty} d(\mu)g(\mu)s_{F(\mu)}(F(R))d\mu dR \tag{20}$$

$$\approx N\int_{-\infty}^{\infty} d(\mu)g(\mu)\int_{\mu}^{\infty} p(R)dRd\mu \tag{21}$$

$$= N\int_{-\infty}^{\infty} d(\mu)g(\mu)(1-P(\mu))d\mu, \tag{22}$$

where P is the cumulative density function of the reward distribution. In Eq. (21), we approximated the sigmoid with a step function at μ. This approximation is close if $p(R)$ is near constant over the range where the sigmoid h_μ changes from 0 to 1. This is the case if the sigmoids are reasonably steep.

There are two further constraints, one to make d a valid probability density and one to limit the range of F to the unit interval:

$$D = \int_{-\infty}^{\infty} d(R)dR = 1, \qquad d(R) > 0 \tag{23}$$

$$F_{\max} = \int_{-\infty}^{\infty} f(R)dR = 1, \qquad f(R) > 0. \tag{24}$$

Solving the optimization. Now we are ready to solve this optimization problem using Lagrange multipliers. To do so, we first compute the following functional derivatives of our objective function:

$$\frac{\delta I}{\delta f}(R) = \frac{p(R)}{f(R)}, \tag{25}$$

$$\frac{\delta I}{\delta g}(R) = \frac{p(R)}{g(R)} \text{ and} \tag{26}$$

$$\frac{\delta I}{\delta d}(R) = \frac{p(R)}{d(R)}. \tag{27}$$

Similarly, we compute the functional derivatives of the constraints. First, for the expected spike rate,

$$\frac{\delta \mathbb{E}[r]}{\delta f}(R) = 0, \tag{28}$$

$$\frac{\delta \mathbb{E}[r]}{\delta g}(R) = (1-P(R))d(R) \text{ and} \tag{29}$$

$$\frac{\delta \mathbb{E}[r]}{\delta d}(R) = (1-P(R))g(R). \tag{30}$$

Second, for the integral of the density D,

$$\frac{\delta D}{\delta f}(R) = 0, \tag{31}$$

$$\frac{\delta D}{\delta g}(R) = 0 \text{ and} \tag{32}$$

$$\frac{\delta D}{\delta d}(R) = 1. \tag{33}$$

Finally, for the upper end of the interval reached by F, $F_{\max}$,

$$\frac{\delta F_{\max}}{\delta f}(R) = 1, \tag{34}$$

$$\frac{\delta F_{\max}}{\delta g}(R) = 0 \text{ and} \tag{35}$$

$$\frac{\delta F_{\max}}{\delta d}(R) = 0. \tag{36}$$

With these derivatives in place, we solve the optimization problem using Lagrange multipliers λ_R, λ_D and λ_F, which results in the following three equations:

$$\frac{p(R)}{f(R)} - \lambda_F = 0, \tag{37}$$

$$\frac{p(R)}{g(R)} - \lambda_R d(R)(1-P(R)) = 0 \text{ and} \tag{38}$$

$$\frac{p(R)}{d(R)} - \lambda_R g(R)(1-P(R)) - \lambda_D = 0. \tag{39}$$

From Eq. (37), we obtain

$$f(R) = \frac{2p(R)}{\lambda_F} \propto p(R). \tag{40}$$

From Eq. (38), we obtain

$$\lambda_R d(R)g(R)(1-P(R)) = p(R) \tag{41}$$

$$d(R)g(R) = \frac{p(R)}{\lambda_R(1-P(R))} \propto \frac{p(R)}{1-P(R)} \tag{42}$$

By inserting this result into Eq. (39), we see that it is guaranteed with $\lambda_D = 0$.

We conclude that under a constraint on the expected number of spikes, the neural population that optimizes the Fisher information bound on mutual information has the following properties:

- The inverse slope of the sigmoids is proportional to the probability density.
- The product of the probability density for neurons with a given midpoint and the gain for neurons with this midpoint is equal to the ratio of the reward probability density and the probability that rewards larger than the midpoint appear.

The formulation by Ganguli and Simoncelli[18] with equally spaced neurons in the unit interval enforces the additional constraint that $d(R) \propto f(R)$ and thus arrives at the special case of $d(R) \propto p(R)$ and $g(R) \propto (1-P(R))^{-1}$.

Splitting density and gain. In our solution, d and g entirely compensate for each other. Due to the fact that their product needs to be proportional to a product of the two factors $p(R)$ and $\frac{1}{1-P(R)}$, a natural choice to parameterize a family of solutions is to use a parameter $\alpha \in [0, 1]$ that trades off the distribution of the two factors to density and gain,

$$d(R) \propto \frac{p(R)}{(1-P(R))^{1-\alpha}} \qquad g(R) \propto \frac{1}{(1-P(R))^{\alpha}}. \tag{43}$$

Effectively, α trades off the gain increase at higher thresholds against placing more neurons at higher thresholds. We illustrate this for several different reward distributions and values of α in Extended Data Fig. 2.

Other ways of writing $\frac{p(R)}{(1-P(R))}$ as a product of neural density and gain are possible, but the split we chose has two convenient properties. First, the density $p(R)$ appears only as a whole factor, which corresponds to the density of the reward distribution actually observed by the animal. This allows realistic learning rules for the neurons, as we derive below. Second, this formula for the density yields a valid density when $\alpha > 0$ independent of the reward distribution so that the same split can be applied to all distributions. We can derive this fact by looking at the uniform distribution on $[0, 1]$. For this reward distribution, the density of neurons becomes $(1-x)^{\alpha-1}$, whose integral over $[0, 1]$ is

$$\int_0^1 (1-x)^{\alpha-1} dx = \left[-\frac{1}{\alpha}(1-x)^{\alpha}\right]_0^1 = \begin{cases} \frac{1}{\alpha}, \text{for } \alpha > 0 \\ \infty, \text{for } \alpha \le 0 \end{cases} \tag{44}$$

Thus, the valid range for α for the uniform distribution is all positive numbers. For any other reward distribution, we can transform the distribution of neurons for the uniform distribution with the inverse cumulative density function P^{-1}. This yields samples with the desired density $\frac{1}{\alpha}\frac{p(R)}{(1-P(R))^{1-\alpha}}$, which must therefore be a valid density.

Choosing a solution. We fitted α using only the midpoints of the measured neurons. To do so, we extracted the midpoints of the measured from logistic function fits as described above. We then estimated α using maximum-likelihood estimation, resulting in $\hat{\alpha} = 0.673$.

The requirements we put forth in Eqs. (6)–(8) are approximately met by many populations of sigmoid-shaped tuning curves. Thus, the exact shape and, in particular, the slope of the sigmoid are not fixed by these efficient coding considerations. For all optimal populations in this paper, we use the cumulative density functions of beta distributions because these sigmoid functions are defined on the unit interval $[0, 1]$ and become exactly 0 and 1 at the ends of the interval, which yields good boundary behavior. These sigmoid functions are not exactly shifted versions of each other (as we assumed in the derivations), but functions that are shifted versions of each other lead to relatively strong boundary effects that produce worse deviations from the theory. To place a neuron at a specific position $p \in [0, 1]$, we set the parameters such that p is the mean of the beta distribution $(a = kp, b = k(1-p))$. Thus, we are left with one more parameter $k > 0$, which controls the slope of the sigmoid functions. As the efficient coding scheme does not determine what the overall slope should be, we set $k = 5.20$ to fit the threshold distribution of measured neurons as described below. We do not expect other shapes to behave fundamentally differently.

We used a fixed spontaneous firing rate r^* for all RPENs. We fitted r^* and the slope parameter k to minimize the mean squared error between the thresholds of the efficient code, $\theta_i^* = h_i^{-1}(r^*)$, and the thresholds of measured neurons,

$$\underset{r^*,k}{\operatorname{argmin}}\, \mathbb{E}\left[\theta_i - h_i^{-1}(r^*;k)\right]^2.$$

The resulting estimates were $\hat{r}^* = 7.63$ and $\hat{k} = 5.20$.

Finally, the efficient code has a parameter $r_{\max}$ that restricts the budget for spikes. In the efficient code, the gain of all neurons is proportional to this parameter. To fit this parameter, we fit sigmoids to both the empirical and the efficient tuning curves as described above and added the squared error of the neuron-averaged gains to the optimization objective.

Checking the solution. We performed several checks to confirm that the derived population is indeed efficient given the reward distribution. To do so, we compared the amount of information communicated to some other populations with the same number of neurons and the same expected number of spikes (Extended Data Fig. 1). The optimized populations retain more information than other populations. The population with the optimized $\alpha = 0.673$ is equally efficient as the solution of Ganguli and Simoncelli[18], which corresponds to $\alpha = 1$.

Other objectives. Our predictions stay qualitatively similar for optimization objectives that transform the Fisher information in different ways, such as the discrimax objective studied by Ganguli and Simoncelli[18]. In particular, our results can be derived whenever the loss is an expected value over a convex function of Fisher information, which goes to infinity at 0 and is monotonically decreasing. Then there are diminishing returns for increasing the Fisher information at any specific reward value, but we need to achieve some nonzero Fisher information for finite loss. The optimal population will then always be a trade-off between distributing the Fisher information broadly and the lower cost for higher thresholds.

As the Fisher information itself and the expected spike rate both depend on the density and gain only through the product of density and gain, we will find a trade-off between density and gain for any objective function defined in this way.

Learning rules for the efficient code

Here, we define the learning rules for the efficient code exactly and show that they converge to the desired population.

Midpoints. We start with a learning rule that makes the midpoints converge to the quantiles of the reward distribution so that their density is proportional to $p(R)$. This covers the $\alpha = 1$ case of Eq. (2). We will then describe how to achieve other α values by choosing the quantiles from a specific nonuniform distribution.

To create a neuron that converges to the quantile q of the reward distribution, we can adjust its midpoint μ in analogy to the distributional RL rule. When a reward above μ occurs, we increase μ by a constant value γ_+; if a reward below μ occurs, we decrease μ by a (usually) different value γ_-:

$$\mu_{i+1} = \begin{cases} \mu_i - \gamma_- & \text{for } R \le \mu_i; \\ \mu_i + \gamma_+ & \text{for } R > \mu_i. \end{cases} \tag{45}$$

The expected change of the midpoint with this learning rule is

$$\mathbb{E}(\mu_{i+1} - \mu_i) = P(R > \mu_i)\gamma_+ - P(R \le \mu_i)\gamma_- \tag{46}$$

$$= (1 - P(R \le \mu_i))\gamma_+ - P(R \le \mu_i)\gamma_- \tag{47}$$

$$= \gamma_+ - P(R \le \mu_i)(\gamma_+ + \gamma_-), \tag{48}$$

which is exactly 0 if the current quantile of μ, $P(R \le \mu_i)$, satisfies

$$P(R \le \mu_i) = \frac{\gamma_+}{\gamma_+ + \gamma_-}, \tag{49}$$

negative if it is larger and positive if it is smaller. If we set γ_+ and γ_- such that $q = \frac{\gamma_+}{\gamma_+ + \gamma_-}$, the neuron with these parameter values will converge to (or oscillate around) the desired quantile. By setting the distribution of γ_+ and γ_- values across the population of neurons, we can thus enable convergence to an arbitrary distribution of quantiles. Choosing a uniform distribution solves the $\alpha = 1$ case.

To enable convergence to a distribution with density proportional to $\frac{p(R)}{(1-P(R))^{1-\alpha}}$ for $\alpha \ne 1$, we can choose quantiles proportional to $\frac{1}{(1-P(R))^{1-\alpha}}$. By the transformation theorem for probability densities, this will result in a distribution with the desired density.

Slopes. The slopes need to converge to values proportional to the local density of rewards (see Eq. (40)). As in kernel density estimation, an estimate of the density around the midpoint of a neuron μ can be found by evaluating a positive kernel function K with integral 1 at the observed reward values R_i,

$$\hat{d}(\mu) = \frac{1}{N}\sum_i K(R_i - \mu). \qquad (50)$$

From this equation, we could already derive a simple learning rule for the slope of the sigmoids, namely setting them based on a running average of these kernel evaluations. However, this would imply a fixed kernel size for any reward distribution, which would be suboptimal if the variance of reward distributions varied. To avoid this problem, we can use a learning rule that adapts the kernel size simultaneously with the slope. We set the width of the kernel h antiproportional to the slope b of the sigmoid function. Thus, the tuning function and the kernel are stretched and shrunk equally such that $bh = \text{constant}$. We then get

$$\text{constant} = bh = \hat{d}(\mu)h = \frac{1}{N}\sum_{i=1}^{N} k\left(\frac{R_i - \mu}{h}\right). \qquad (51)$$

In other words, we need to define a learning rule such that $k\left(\frac{R_i-\mu}{h}\right)$ converges to a constant. This can be achieved for any constant $c \in [0, k(0)]$ by changing the slope and kernel width proportional to $k\left(\frac{R_i-\mu}{h}\right) - c$. As h and b need to stay positive, we perform these updates in log-space, that is, multiply both by $\exp \lambda\left(k\left(\frac{R_i-\mu}{h}\right) - c\right)$ for some small learning rate λ.

Gain learning. As the first of two ways of adjusting the gain of a neuron, we can use the expected firing rate before scaling with the gain. This can achieve convergence to the efficient code because the factor $1 - P(\mu)$ arises as an approximation of the expected firing rate of a neuron in Eq. (22) before multiplication with the gain. To learn this value, a neuron can compute a running average of the unscaled neural responses $\bar{r}$ with some learning rate λ_r:

$$\bar{r}_{i+1} = (1 - \lambda_r)\bar{r}_i + \lambda_r \frac{r_i}{g_i}. \qquad (52)$$

Plugging this into the efficient code solution yields

$$g_i = \frac{1}{\bar{r}_i^{\alpha}}. \qquad (53)$$

With $\alpha = 1$, this learning rule would yield the same expected firing rate for all neurons. With $\alpha < 1$, it yields a less extreme growth in gain.

Gain coupled to quantile. Another way of adjusting the gain is by setting a fixed gain based on the quantile a neuron is meant to converge to. As the quantile fixes $P(\mu)$, we can read off the correct value from the analytical efficient coding solution, that is, set the gain g equal to $\frac{1}{(1-q)^{\alpha}}$ for neurons that are meant to converge to the quantile q.

Expectiles instead of quantiles

For the figures in the main text, we placed neuron midpoints at the quantiles of the reward distribution as predicted by maximizing coding efficiency. As distributional RL predicts that neurons should converge to expectiles instead, we tested the efficiency of such a code in Extended Data Fig. 6. The efficiency of the code is slightly lower for the expectile-based code due to a slight shift of midpoints toward the mean of the distribution. These differences are small, and the code does not show qualitative differences. We understand them as essentially equivalent.

Comparing thresholds and midpoints

Because we are using two metrics for the placement of neurons, threshold and midpoint, we discuss here how they compare. In our formalism, 'density' refers to the midpoint of a neuron with a sigmoid response function. Threshold is the reward when a neuron fires at spontaneous activity, this is, the point at which the neuron codes zero prediction error in RL. Their distributions are slightly different. The relationship between the two depends on the relationship between the gain of the neuron (which defines the midpoint) and its spontaneous firing rate (which defines the threshold). A neuron whose gain exceeds twice its spontaneous firing rate has a higher midpoint than threshold, as reward needs to be higher for the firing rate to reach half the gain than to reach the spontaneous rate. The threshold will then lie in the lower convex part of the response function. Correspondingly, a neuron with gain less than twice its spontaneous rate has a higher threshold than midpoint, and its threshold will lie in the upper concave part of the response function. In the efficient code, gain increases with threshold. As a result, the prediction is that low thresholds are higher than their corresponding midpoints, and high thresholds are lower than their corresponding midpoints. In turn, this means that the threshold distribution should be less dispersed than the midpoint distribution.

We illustrate this in the variable-magnitude task (Extended Data Fig. 4). Empirically, the threshold distribution is indeed narrower (and taller) than the midpoint distribution, as predicted by the efficient code. Additionally, the spontaneous firing rate we estimated for this task is lower than the average half gain. Correspondingly, the average threshold (5.00 µl) is smaller than the average midpoint (5.96 µl). However, these shifts are all relatively small, and the relationships with other variables (Fig. 2b,d,f) hold for midpoints as they do for thresholds.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

No new data were measured for this project. The data collected by Eshel et al.[15] that we analyze here were kindly made available by Dabney et al.[16] at https://doi.org/10.17605/OSF.IO/UX5RG.

Code availability

The code to recreate our analyses[42] is available at https://github.com/dongjae-kim/efficient-coding-dist-rl.

References

37. Wei, X.-X. & Stocker, A. A. A Bayesian observer model constrained by efficient coding can explain 'anti-Bayesian' percepts. *Nat. Neurosci.* **18**, 1509–1517 (2015).
38. Brunel, N. & Nadal, J.-P. Mutual information, Fisher information, and population coding. *Neural Comput.* **10**, 1731–1757 (1998).
39. Cover, T. M. & Thomas, J. A. *Elements of Information Theory* (Wiley, 1991).
40. Wei, X.-X. & Stocker, A. A. Mutual information, Fisher information, and efficient coding. *Neural Comput.* **28**, 305–326 (2016).
41. Bethge, M., Rotermund, D. & Pawelzik, K. Optimal short-term population coding: when Fisher information fails. *Neural Comput.* **14**, 2317–2351 (2002).
42. Schütt, H., Kim, D. & Ma, W. J. Code for efficient coding and distributional reinforcement learning. *Zenodo* https://doi.org/10.5281/zenodo.10669061

Acknowledgements

We thank H.-H. Li for valuable discussions. We received no specific funding for this work.

Author contributions

H.H.S. derived the efficient code. H.H.S. and D.K. analyzed the neural data. W.J.M. supervised the project. All authors wrote the manuscript.

Competing interests

The authors declare no competing interests.

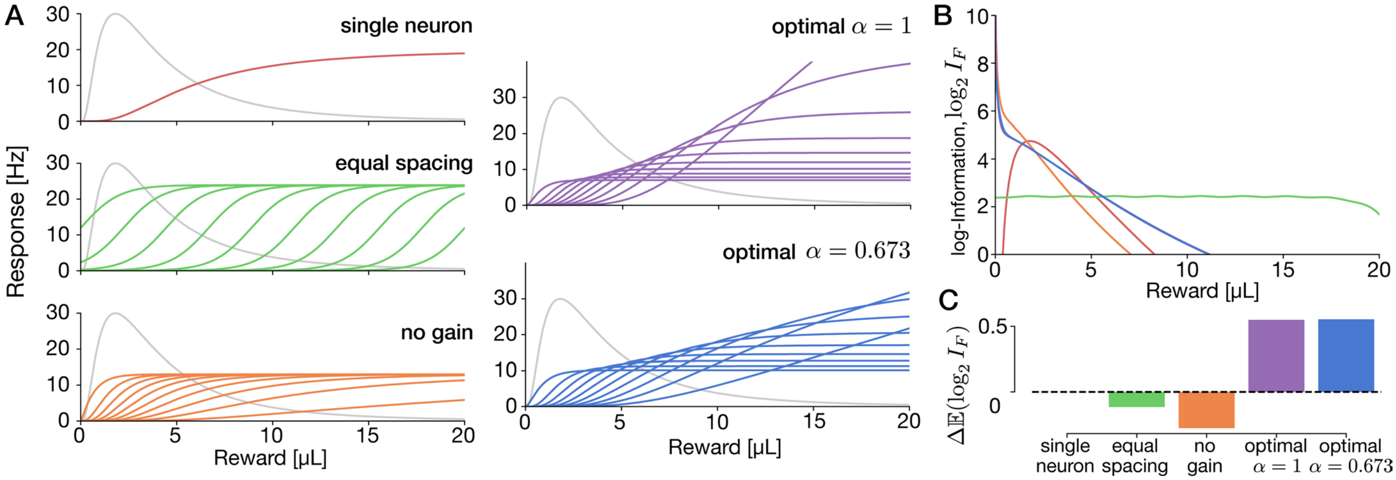
Additional information

Extended data is available for this paper at https://doi.org/10.1038/s41593-024-01671-x.

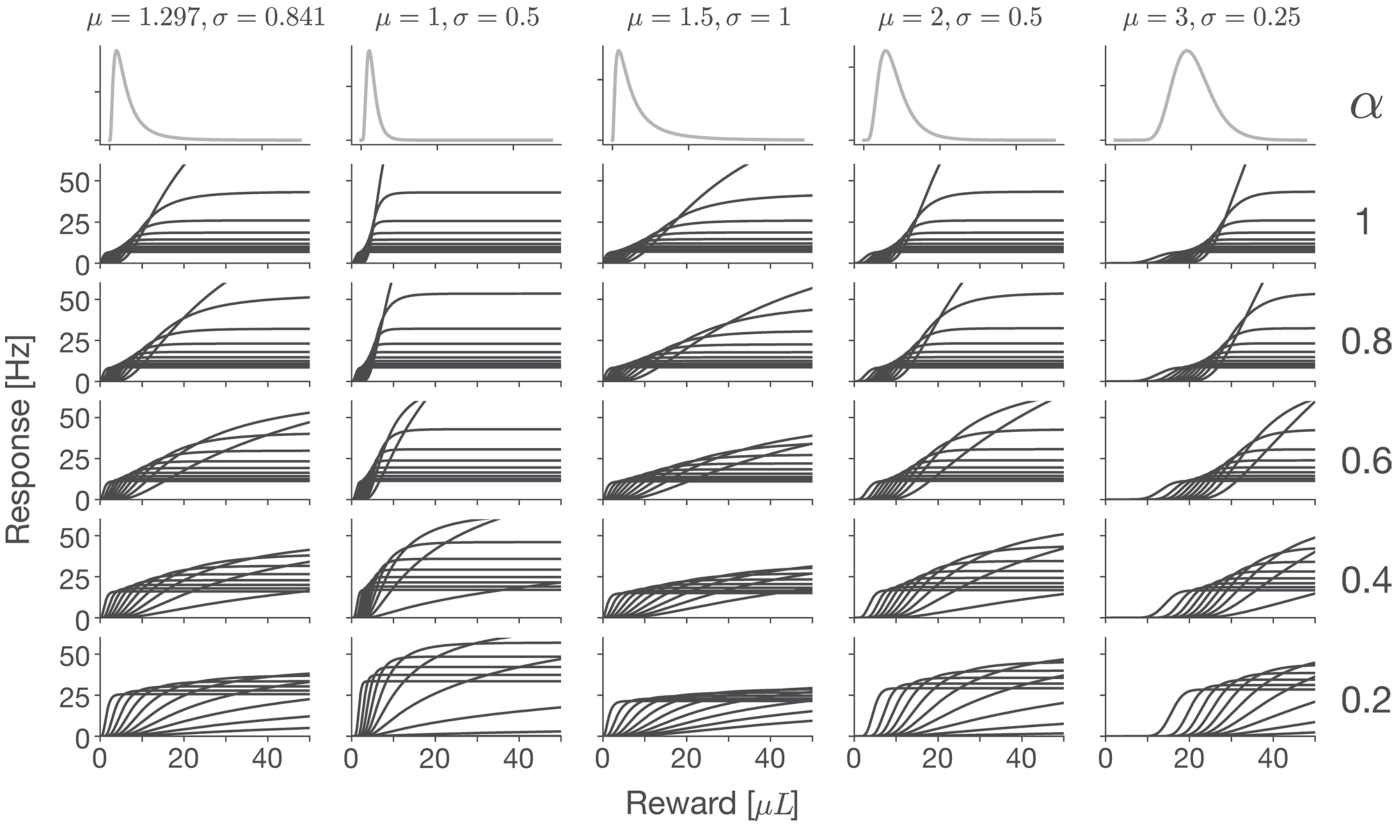
Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41593-024-01671-x.

Correspondence and requests for materials should be addressed to Heiko H. Schütt.

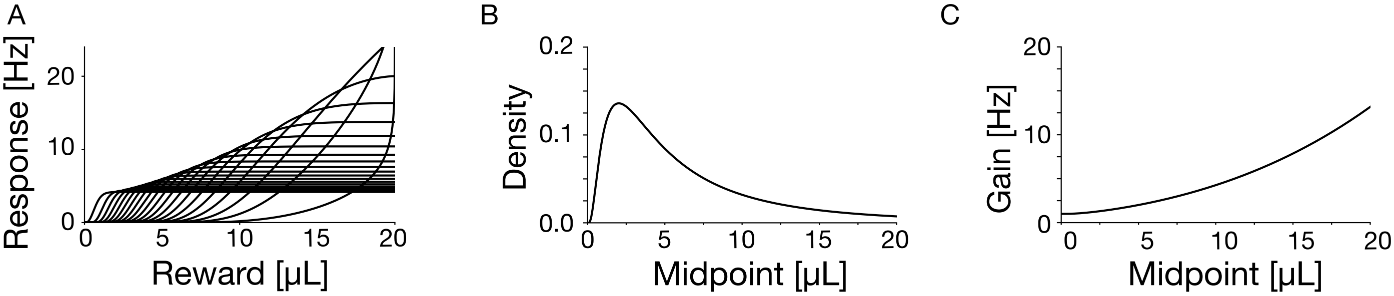
Peer review information *Nature Neuroscience* thanks the anonymous reviewers for their contribution to the peer review of this work.



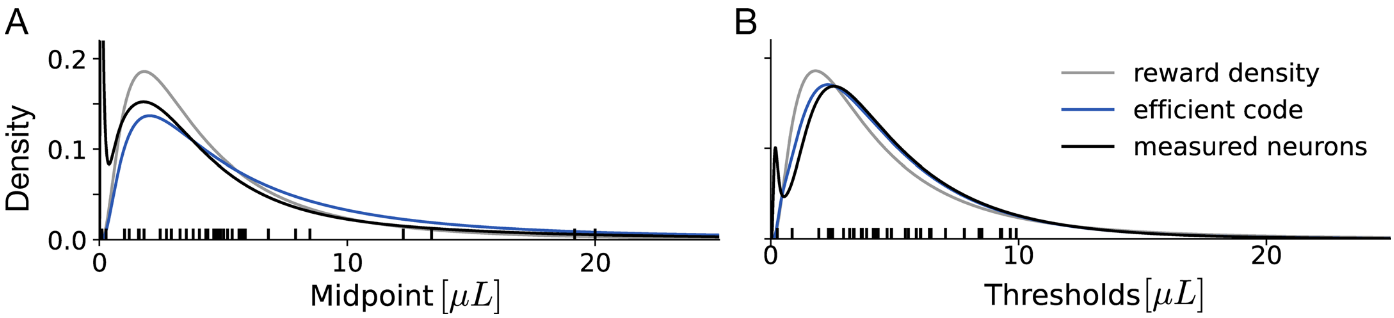
Extended Data Fig. 1 | Comparing encoding populations for reward with 10 neurons and the same expected number of spikes. A: Compared neuronal populations: **single neuron**: All neurons share the same response curve, optimized to maximize transferred information. **equal spacing**: neurons tile the space, not optimized. **no gain**: positions and slopes are optimized, but all neurons have equal gain. **optimal** $\alpha = 1$: fully optimized population as derived previously[18] with density proportional to the distribution. **optimal** $\alpha = 0.673$: Equally optimal distribution but with α fit to match the midpoint distribution for the optimal code and the experimental data. **B**: Fisher information as a function of reward for each of the populations. **C**: Expected logarithm of Fisher information under the reward distribution relative to the single-neuron case.



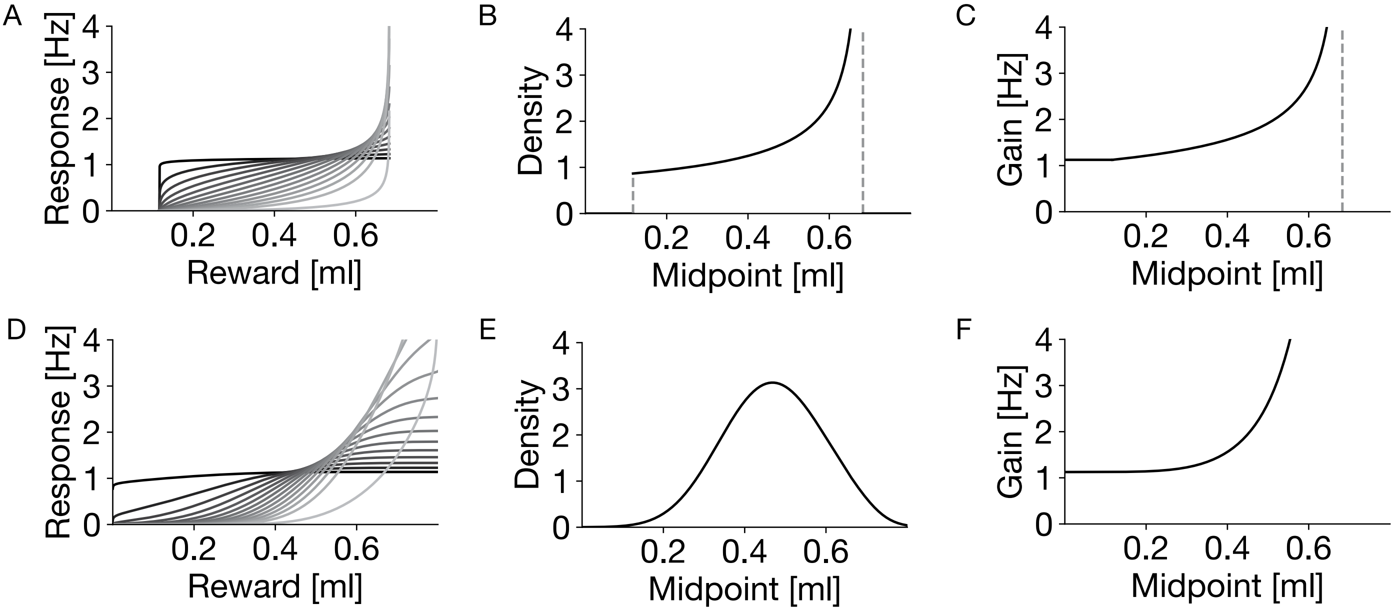
Extended Data Fig. 2 | Illustration of the solution of the efficient coding problem, varying *α* (rows) and the reward distribution (columns). The reward distributions are all log-normal distributions with their pdfs and parameters plotted at the top.



Extended Data Fig. 3 | Efficient code for the variable-reward task[14]**.** **A**: Tuning curves. For clarity, only 20 of 39 neurons are shown. **B**: Density of neurons as a function of midpoint. **C**: Gain as a function of midpoint.

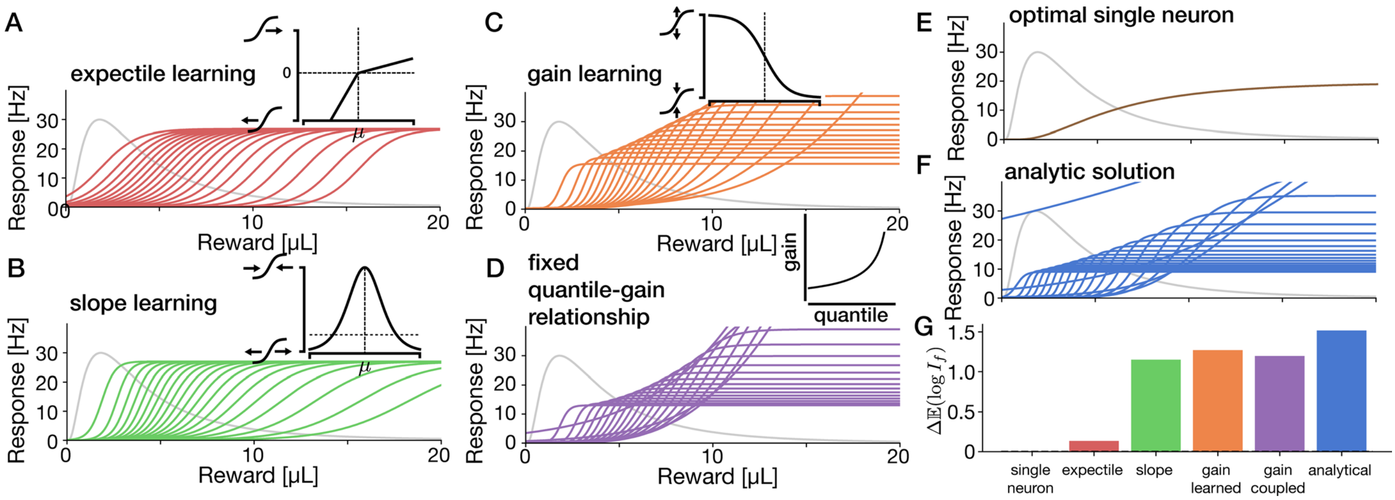


Extended Data Fig. 4 | Log-normal kernel density estimation of midpoints and threshold. A: Midpoints. **B**: Thresholds. Measured neurons (black) and efficient code (cyan) are overlayed over the reward density (gray).



Extended Data Fig. 5 | Efficient code for the variable-magnitude task[17]**. A-C**: Efficient code for the uniform distribution. **D-F**: Efficient code for the normal distribution. **A,D**: Tuning curves. For clarity, only 13 of 40 neurons are shown. **B,E**: Density. **C,F**: Gain.

https://doi.org/10.1038/s41593-024-01671-x



Extended Data Fig. 6 | Evaluation of learning rules placing neurons' midpoints at expectiles instead of quantiles. Plotting conventions as in Fig. 4. Each panel shows the converged population of 20 neurons after learning based on 20, 000 reward presentations. The inset illustrates the learning rule. **A**: Learning the position on the reward axis for the neurons to converge to the quantiles of the distribution. This learning rule is the distribution RL learning rule. **B**: Additionally learning the slope of the neurons to be proportional to the local density by increasing the slope when the reward falls within the dynamic range and decreasing otherwise. **C**: First method to set the gain: iterative adjustment to converge to a fixed average firing rate. **D**: Second method to set the gain: use a fixed gain per neuron based on the quantile it will eventually converge to. **E**: The efficient tuning curve for a single neuron. **F**: The analytically derived optimal solution. **G**: Comparison of information transfer across the different populations with the same number of neurons and expected firing rate.

nature portfolio

Corresponding author(s): Wei Ji Ma

Last updated by author(s): Feb 16, 2024

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed	
☐	☒	The exact sample size (n) for each experimental group/condition, given as a discrete number and unit of measurement
☐	☒	A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒	The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐	A description of all covariates tested
☐	☒	A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒	A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☐	☒	For null hypothesis testing, the test statistic (e.g. F, t, r) with confidence intervals, effect sizes, degrees of freedom and P value noted *Give P values as exact values whenever suitable.*
☒	☐	For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐	For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☐	☒	Estimates of effect sizes (e.g. Cohen's d, Pearson's r), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	All data was precollected. We used matlab and python to load and analyze these data.
Data analysis	The code to recreate our analyses is available at https://github.com/dongjae-kim/efficient-coding-dist-rl

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our policy

No new data was measured for this project. The data collected by Eshel et al. (2016) that we analyze here are kindly made available by Dabney et al. (2020) at https://doi.org/10.17605/OSF.IO/UX5RG.

Research involving human participants, their data, or biological material

Policy information about studies with human participants or human data. See also policy information about sex, gender (identity/presentation), and sexual orientation and race, ethnicity and racism.

Reporting on sex and gender	N/A
Reporting on race, ethnicity, or other socially relevant groupings	N/A
Population characteristics	N/A
Recruitment	N/A
Ethics oversight	N/A

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	40 neurons from 5 mice. 40 neurons from 2 monkeys. All data we use here were already publicly available. Thus, no statistical method was used by us to predetermine sample size.
Data exclusions	One neuron from the mice was excluded because it had never shown a single response larger than its baseline for any reward magnitude following the analysis decisions of the original authors
Replication	We did not attempt to replicate any of the data we analyze.
Randomization	The experiments were not randomized, as there were no experimental groups to assign to randomly.
Blinding	The investigators were blind to our hypothesis as this hypothesis was not stated yet, when the data were recorded, but not to any aspect of the experimental design. Data collection and analysis were not performed blind to the conditions of the experiments.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology and archaeology
☒	☐ Animals and other organisms
☒	☐ Clinical data
☒	☐ Dual use research of concern
☒	☐ Plants

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging

Plants

Seed stocks	N/A
Novel plant genotypes	N/A
Authentication	N/A