Towards a unified theory of efficient, predictive and sparse coding: Supplementary information

Matthew Chalk, Olivier Marre, Gasper Tkacik

1 Efficient coding models

The efficient coding hypothesis posits that sensory systems have evolved to transmit maximal information about incoming sensory signals, given internal resource constraints (such as internal noise, and/or metabolic cost) [1, 2, 3, 4, 5, 6, 7]. It has been successful in predicting a host of different neural response properties from first principles. Nonetheless, there is often confusion in the literature, due to the fact that different authors have made very different assumptions about: (i) what sensory information is relevant (and thus, *should* be encoded); (ii) the internal constraints (determining what information *can* be encoded).

In the following we provide a brief (non-exhaustive) overview of the various types of efficient coding model that have been proposed (illustrated in SI Fig 1), so as to clarify the relation between them.

1.1 Redundancy reduction

Efficient coding models have usually assumed that the goal of sensory processing is to encode maximal information about *all* incoming signals, given internal constraints. In the low-noise limit, this implies that neurons should remove redundancies in their inputs, to achieve statistically independent responses [5, 6, 7, 8, 9].

Considering, for the moment, only second-order statistics, redundancy reduction implies that neurons should whiten incoming signals, to achieve decorrelated responses. Previous work showed that this can explain many aspects of low-level visual neuron responses, such as the centre-surround receptive fields (RFs) of neurons in the retina [3, 10], and the temporal filtering properties of neurons in the LGN [11].

More generally, to achieve independent responses, neurons must also remove high-order (i.e. beyond covariance) statistical redundancies in their inputs. One way to do this is via 'sparse coding', where only a small proportion of neurons are active at any one time [12]. Indeed, given certain assumptions about the statistical structure of sensory signals (i.e. that they are generated by linearly combing a set of independent sparsely distributed features), maximising the sparsity of neural responses is equivalent to maximising their independence [13, 14, 15]. In a seminal paper, Olshausen & Field showed that learning a sparse code of natural images results in local orientated filters, that closely resemble the RFs of V1 simple cells [16]. Since then, sparse coding has been used to model several other aspects of low-level visual neuron responses [17], in addition to coding by auditory [18] and olfactory [19] neurons.

It has been proposed that statistically independent responses could be achieved if sensory neurons encode a 'prediction error' equal to the difference between their input, and an internal prediction generated by the network [10, 20]. This could be implemented in a hierarchical network, with feed-forward signals transmitting an error signal, while feed-back signals transmitting a prediction [20]. Alternatively, recent works have shown how predictive coding could be implemented within a single densely connected recurrent spiking network [21, 22].

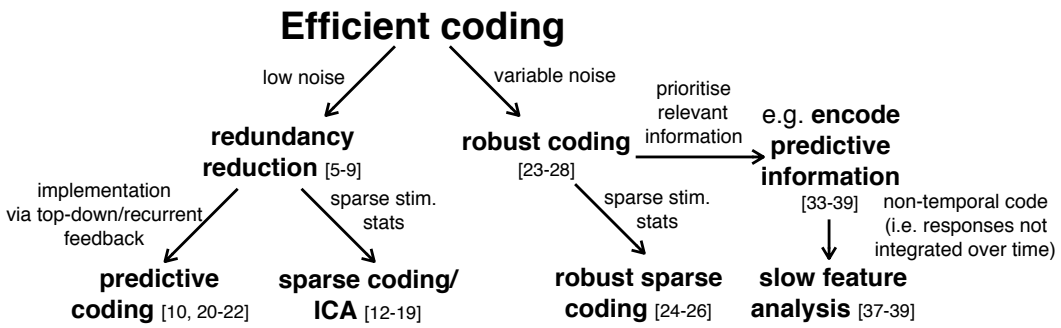

Figure 1: Schematic of various types of efficient coding model, with corresponding references.

1.2 Robust coding

The mutual information between responses, R, and stimulus X, can be expressed as: $I(R;X) = H(R) - H(R|X)$, where $H(R)$ is the response entropy, and $H(R|X)$ is the noise entropy. At low-noise, where the second, noise entropy term is negligible, information is maximised by maximising the response entropy, $H(R)$ (via redundancy reduction). At higher noise, it becomes important to minimise the noise entropy, $H(R|X)$, leading to qualitatively different predicted neural responses [4].

In a seminal paper, Atick & Redlich showed that varying the signal-to-noise ratio leads to a qualitative change in the predicted neural code, with neurons whitening their inputs at low noise (to minimise redundancy) and smoothing their inputs at high noise (to average the noise) [23]. Interestingly, their model was able to explain how the shape of retinal ganglion cell (RGC) RFs vary with visual contrast.

More recently, several authors proposed 'robust coding' models detailing how 'sensory noise' (added to the sensory input), and 'neural noise' (added to the neural responses) alter the predicted neural responses [24, 25, 26, 27]. These models were able to account for various further aspects of RGC responses, including how the shape and overlap of RGC RFs varies with visual eccentricity. Further, they showed how sparse coding varies with the amplitude of neural and sensory noise. For example, Karklin & Simoncelli [26] showed that at low noise enforcing sparsity leads to local orientated spatial filters (as in [16]), while at high noise it leads to circularly symmetric spatial filters.

Tkacik et al. studied how the recurrent connectivity of an efficient spiking network should vary with the signal-to-noise ratio [28]. Interestingly, they found that at low signal-to-noise, information maximisation predicts an attractor like structure of the neural code. While highly redundant, this type of code allows the network to mitigate the effects of noise.

Given time-varying stimulus statistics time, the speed that neurons can adapt to efficiently encode their inputs is limited by the need to collect new statistics. Interestingly this was observed for motion-sensitive neurons (H1) neurons in the fly visual system, which not only adapt to efficiently encode new input statistics [29], but whose speed of adaptation approaches the physical limits imposed by statistical sampling and noise [30].

1.3 Coding relevant information

An alternative hypothesis is that, rather than encoding *all* sensory signals, neural circuits preferentially encode *behaviourally relevant* signals. Indeed, Machens et al. found that grasshopper auditory neurons are optimised to efficiently encode behaviourally relevant vocalisation signals, rather than the sound signals most commonly found in their environment [31]. Further in higher-level sensory areas, many neurons are specialised for encoding features that are behaviourally relevant (e.g. faces) [32], rather than features that are statistically likely (e.g. clouds).

A difficulty here is that, except in special cases, it is hard to know which sensory information is relevant to an organism. To overcome this, Bialek & colleagues proposed that a minimal criterion for a

stimulus to be behaviourally relevant, is that it can be used to predict what will happen in the future. This led them to hypothesise that sensory neural circuits are set up to encode maximal information about stimuli that are predictive about the future, given a constraint on the information encoded about previous inputs [33, 34, 35].

While intriguing, there is currently little theoretical work exploring the neural implications of this idea. Further, previous work only considered a highly restrictive scenario where neurons are assumed to encode information redundantly, via their instantaneous responses [35, 36, 37]. Interestingly, Creutzig et al. [37] showed that, in this case (and given some further assumptions, such as linear gaussian stimulus statistics) efficiently encoding the future is equivalent to 'slow feature analysis' (SFA), a method for extracting slowly varying components from quickly varying input signals, used previously to account for the response properties of complex cells in area V1 [38, 39].

2 General framework

We consider a stimulus represented by the time series, $\{\ldots, Y_{t-1}, Y_t\}$, which is corrupted by additive gaussian white noise to produce a input, $\{\ldots, X_{t-1}, X_t\}$, received by a population of sensory neurons. We ask what is the optimal neural code, $p(R_t|X_{-\infty:t})$, such that responses in a time window from $t-\tau$ to t encode maximal information about the stimulus between time $t+\Delta_1$ and $t+\Delta_2$, constrained on the total information encoded about previous inputs, up to time t. This can be achieved by maximising the following 'information bottleneck' (IB) objective function [40]:

$$L_{p(R_t|X_{-\infty:t})} = I\left(R_{t-\tau:t}; Y_{t+(\Delta_1:\Delta_2)}\right) - \gamma I\left(R_{t-\tau:t}; X_{-\infty:t}\right) \tag{1}$$

The first term denotes the mutual information between $R_{t-\tau:t}$ and $Y_{t+(\Delta_1:\Delta_2)}$, to be maximised, and the second term denotes the mutual information between $R_{t-\tau:t}$ and $X_{-\infty:t}$, to be constrained. A constant, γ, determines the strength of this constraint, and thus, the tradeoff between coding fidelity and compression.

The above objective function is valid for modeling predictive coding, when $\Delta_1 > 0$ & $\Delta_2 > 0$. However, we wanted a framework that would: (i) give non-trivial solutions for *all* Δ_1 and Δ_2; (ii) allow comparison with previous efficient coding models. That the first criterion is not satisfied by the above objective function can be seen by setting $[\Delta_1, \Delta_2] = [-\infty, 0]$ and $Y = X$, in which case the two terms of equation 1 are proportional, and the maximisation is unconstrained.

To overcome this, we considered an alternative objective function:

$$L_{p(r_t|x_{-\infty:t})} = I\left(R_{t-\tau:t}; Y_{t+(\Delta_1:\Delta_2)}\right) - \gamma\tau I\left(R_t; X_{-\infty:t}\right) \tag{2}$$

where we replaced $R_{t-\tau:t}$ in the second, constraint term with the instantaneous response, R_t. If the responses at each time point are conditionally independent, this expression gives a lower bound on the previous IB objective function. Further, when $[\Delta_1, \Delta_2] = [-\infty, 0]$ and $X = Y$, maximising $\tilde{L}$ is equivalent to minimising the temporal redundancy of neural responses (and exactly the same, when $\gamma = 1$). Thus the objective function is equally applicable for modeling efficient coding of past inputs (Δ_1 & $\Delta_2 < 0$) and predictive coding of future inputs (Δ_1 & $\Delta_2 > 0$).

Finally, note that, while in general the decoding window could be of arbitrary length, to limit the number of free parameters in our analysis, we considered the case where $\Delta_1 = \Delta_2$, so that the decoding window is limited to a single time-bin of lag Δ. Setting $X = Y$ (i.e. zero external noise), gives the objective function shown in equation 1 in the main text.

After performing these simplifications, the objective function can be expanded as follows:

$$L_{p(r_t|x_{-\infty:t})} = \langle \log p\left(y_{t+\Delta}|r_{t-\tau:t}\right) + \gamma\tau \log p\left(r_t\right) - \gamma\tau \log p\left(r_t|x_{t-\infty:t}\right)\rangle_{p(r,x,y)} \tag{3}$$

where for notational simplicity, we have omitted the constant stimulus entropy term. Unfortunately, in many cases, this objective function cannot be computed tractably. Instead, we can compute an

approximate lower bound $\tilde{L} < L$, that can be evaluated tractably. To do this, we replace the distributions $p(y_{t+\Delta}|r_{t-\tau:t})$ and $p(r_t)$ with approximate distributions, $q(y_{t+\Delta}|r_{t-\tau:t}) \in Q_{Y|R}$ and $q(r_t) \in Q_R$ (where $Q_{R|X}$ and Q_R denote parametric families of distributions, for which the expectations can be computed tractably) [41]. We then maximise the resulting lower bound $\tilde{L}$, via alternate updates on $p(r_t|x_{-\infty:t})$, $q(r_t)$ and $q(y_{t+\Delta}|r_{t-\tau:t})$.

2.1 Model description

We considered a linear encoding model, with neural responses sampled from a multivariate gaussian, $\mathcal{N}(r_t|\mu_t, \Sigma)$. The mean response, μ_t was obtained by linearly filtering the stimulus, $\mu_t = \sum_{k=1}^{\tau_w} W_k x_{t-k+1}$, where W_k is an $N_r \times N_x$ matrix, denoting the spatial encoding filter at lag k. Σ is an $N_r \times N_r$ symmetric noise covariance matrix. N_r and N_x denote the number of neurons and stimulus dimensions, respectively.

As described above, to formulate a tractable lower bound for the IB objective function, we had to approximate the decoding distribution, $p(y_{t+\Delta}|r_{t-\Delta:t})$, and the response distribution, $p(r_t)$.

We approximated the decoding distribution with a linear gaussian model, $q(y_{t+\Delta}|r_{t-\Delta:t}) = \mathcal{N}(y_{t+\Delta}|\hat{y}_{t+\Delta}, \Lambda)$, with mean, $\hat{y}_{t+\Delta}$, obtained by linearly filtering the responses according to, $\hat{y}_{t+\Delta} = \sum_{k=1}^{\tau} U_k r_{t-k+1}$, where U_k is an $N_x \times N_r$ matrix, denoting the decoding filter at lag k. Λ is an $N_x \times N_x$ symmetric error covariance matrix.

Previous work has shown that efficient coding of natural stimuli can be achieved via a 'sparse' code, where individual neurons are selective for rarely occuring (i.e. sparse) stimulus features. To allow for sparse coding solutions in our framework, we approximated the response distribution $p(r_t)$ using a student-t distribution, $q(r_t) = \prod_{i=1}^{N_r} \text{Student}\left(r_{i,t}|0, \omega_i^2, \nu_i\right)$, where $r_{i,t}$ denotes the response of the i^{th} neuron at time t, and ω_i^2, and ν_i are the scale and shape parameters of the student-t distribution, respectively. For the initial simulations with gaussian stimulus statistics, shown in fig. 2, we considered the limit where $\nu_i \to \infty$ (i.e. where $q(r_t)$ is gaussian). In later simulations, shape parameters for each neuron, ν_i, were learned from data.

2.2 Optimisation algorithm

Parameters of the encoding distribution (W & Σ), decoding distribution (U & Λ) and response distribution (ω & ν) were learned using a variational IB algorithm, as described in [41]. First we initialise the parameters of the encoding distribution W, and Σ. Next we perform recursive updates of the decoding distribution (U & Λ) and response distribution (ω & ν) parameters, followed by updates of the encoding distribution (W & Σ). This sequence is repeated until the parameters converge. As a full derivation is given in [41], here we restrict ourselves to describing how each of the model parameters are updated on each iteration of the algorithm.

Decoding distribution. On each iteration, the parameters of the decoding distribution, $q(y_{t+\Delta}|r_{t-\tau:t}) = \mathcal{N}(y_{t+\Delta t}|U r_{t-\tau:t}, \Lambda)$ are updated according to:

$$U \leftarrow C_{y_{t+\Delta}r} C_{rr}^{-1} \qquad \Lambda \leftarrow C_{y_{t+\Delta}y_{t+\Delta}} - C_{y_{t+\Delta}r} U^T \tag{4}$$

where U is an $N_y \times \tau N_r$ matrix, $U = (U_0, \cdots, U_{\tau-1})$. C_{rr} is an $[\tau N_r \times \tau N_r]$ covariance matrix, $C_{rr} = \left\langle \begin{pmatrix} r_t \\ \vdots \\ r_{t-\tau} \end{pmatrix} (r_t, \ldots r_{t-\tau}) \right\rangle$ and $C_{y_{t+\Delta}r}$ is an $[\tau N_y \times \tau N_r]$ covariance matrix, $C_{y_{t+\Delta}r} = \langle y_{t+\Delta}(r_t, \ldots r_{t-\tau}) \rangle$.

Response distribution. As stated earlier, we approximated the marginal response distribution by a student-t distribution, $q(r_t) = \prod_{i=1}^{N_r} \text{Student}\left(r_{i,t}|0, \omega_i^2, \nu_i\right)$, with shape and scale parameters for each

neuron ν_i, and ω_i, respectively. Substituting $q(r_t)$ into the second term of equation 3, gives:

$$\langle \log q\left(r_t\right)\rangle = -\sum_{i=1}^{N_r} \frac{\nu_i+1}{2}\left\langle \log\left(1+\frac{r_{i,t}^2}{\omega_i^2\nu_i}\right)\right\rangle - \frac{1}{2}\log\omega_i^2 + f(\nu_i), \tag{5}$$

where the summation is taken over N_r neurons, and $f(\nu_i) = \log\Gamma\left(\frac{\nu+1}{2}\right) - \log\Gamma\left(\frac{\nu}{2}\right)$. Parameters, ν_i, and ω_i, are updated on each iteration, to maximise $\langle \log q\left(r_t\right)\rangle$. Note that, for non-sparse, gaussian stimuli, the IB algorithm returns $\nu_i \to \infty$ and $\omega_i^2 = \left\langle r_{i,t}^2\right\rangle$, in which case $\langle \log q\left(r_{i,t}\right)\rangle = -\frac{1}{2}\log\left\langle r_{i,t}^2\right\rangle + \text{const.}$, as in equation 3 of the main text.

Unfortunately, the expectation shown above cannot be evaluated in closed form. Instead, we use a variational approximation to construct a lower bound that can be tractably maximised. Following this procedure, as detailed in [41], the scale parameter is updated on each iteration according to,

$$\omega_i^2 \leftarrow \left\langle \xi_{ti} r_{ti}^2 \right\rangle, \tag{6}$$

where $\left\langle r_{ti}^2\right\rangle$, denotes the mean-squared response of the i^{th} neuron at time t, and ξ_{ti} is an additional variational parameter updated on each trial according to: $\xi_{ti} = \frac{\nu_i+1}{\nu_i + \left\langle r_{ti}^2\right\rangle/\omega_i^2}$. (Note that in the limit where $\nu_i \to \infty$, so that the response distribution is near gaussian, $\xi_{ti} = 1$, and $\omega_i^2 = \left\langle r_{ti}^2\right\rangle$.)

The shape parameter, ν_i, is found numerically on each iteration by solving:

$$\psi\left(\frac{\nu_i}{2}\right) - \log\left(\frac{\nu_i}{2}\right) = 1 + \psi(a_i) - \log a_i + \left\langle \log\xi_{ti} - \xi_{ti}\right\rangle, \tag{7}$$

where $\psi(\cdot)$ is the digamma function, and $a_i = \frac{1}{2}\left(\nu_i^{old}+1\right)$.

Encoding distribution. The encoding distribution is described by: $p(r_t|x_{t-\tau_w:t}) = \mathcal{N}\left(r_t|W x_{t-\tau_w:t}, \Sigma\right)$. On each trial, the noise covariance is updated according to:

$$\Sigma^{-1} \leftarrow \frac{1}{\gamma}\sum_{k=0}^{\tau-1} U_k^T \Lambda^{-1} U_k + \Omega^{-1}\left\langle \Xi_t\right\rangle, \tag{8}$$

where U_k denotes the decoding filter at lag k, and Ω and Ξ_n are $N_r \times N_r$ diagonal covariance matrices with diagonal elements $\Omega_{ii} = \omega_i^2$, and $\left(\Xi_t\right)_{ii} = \xi_{ti}$, respectively.

The update for W is given by:

$$w \leftarrow \left(H_f + \gamma H_p\right)^{-1} b \tag{9}$$

where w is an $[N_x\tau_w N r \times 1]$ vector defined by $w = \text{vec}\begin{pmatrix} W_1^T \\ \vdots \\ W_{\tau_w}^T \end{pmatrix}$.

To express H_p and H_f, we start by defining the time series, $(\cdots, z_{t-1}, z_t)$, where $z_t = \begin{pmatrix} x_t \\ x_{t-1} \\ \vdots \\ x_{t-\tau_w+1} \end{pmatrix}$.

H_p is then defined as an $[N_x N r \times N_x N_r]$ square matrix, $\begin{pmatrix} \left(H_p\right)_{11} & 0 & \cdots \\ 0 & \left(H_p\right)_{22} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$ where $\left(H_p\right)_{ii} = \frac{1}{\omega_i^2 i}\left\langle \xi_{i,t} z_t z_t^T\right\rangle$.

H_f is an $[N_x N_r \times N_x N_r]$ square matrix, defined by $\begin{pmatrix} \left(H_f\right)_{11} & \left(H_f\right)_{12} & \cdots \\ \left(H_f\right)_{21} & \left(H_f\right)_{22} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$ where $\left(H_f\right)_{ij} = \sum_{k=1}^{\tau}\sum_{m=1}^{\tau} u_{ki}^T \Lambda^{-1} u_{mi}\left\langle z_{t-k+1} z_{t-m+1}^T\right\rangle$, and u_{ki} is the i^{th} column of the $[N_y \times N_r]$ matrix, U_k.

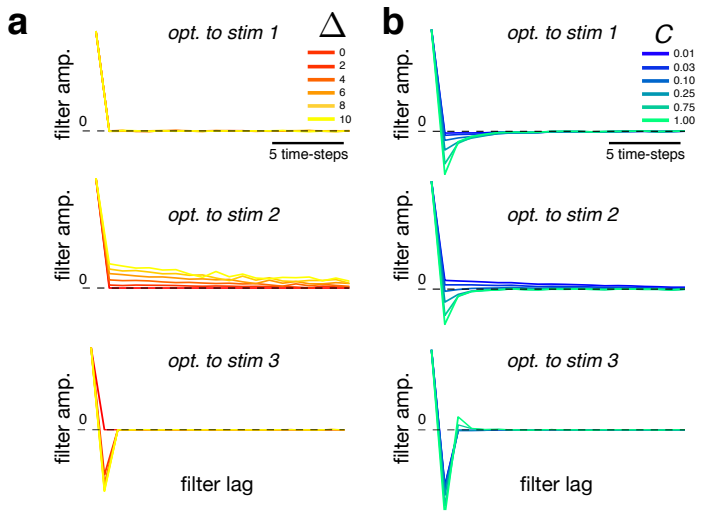

Figure 2: **Dependence of encoding filters on decoding lag, Δ, code length, τ, and coding capacity, C. (a)** Encoding filters after optimised with varying Δ, and $\tau = 0$. Encoding filters are normalised to have the same value at lag 0. **(b)** Same as a, but with filters optimised with $\tau \gg 0$. Plots correspond to filters at varying coding capacity, C, & fixed decoding lag ($\Delta = 3$).

Finally, b is an $[N_x N_r \times 1]$ vector, defined by $b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \end{pmatrix}$, where $b_i = \sum_{k=1}^{\tau} \left\langle z_{t-k+1} y_{t+\Delta}^T \right\rangle \Lambda^{-1} u_{ki}$

.

3 Methods for simulations in the main text

3.1 Neural coding of 1-d gaussian time series

For the initial simulations, shown in figure 2 in the main text, we considered three different 1-d time series. Stimulus 1 ('markov stim.') was generated from an AR1 process, that evolved in time according to the recurrence relation: $x_t = ax_{t-1} + b\eta_t$, where η_t is drawn from a standard normal distribution, and $a = 0.89$ and $b = 0.48$. Stimulus 2 ('two timescales') was constructed from two AR1 series, summed according to, $x_t = \rho x_t^{\text{slow}} + \sqrt{1-\rho^2} x_t^{\text{fast}}$, with $\rho = 0.47$. x^{slow} and x^{fast} were both generated from an AR1 process, with parameters $a = 0.97, b = 0.23$, and $a = 0.67, b = 0.73$, respectively. Stimulus 3 ('inertial') was generated from an AR2 process. The stimulus at time t was given by $x_t = ax_{t-1} + bx_{t-2} + c\eta_t$, where $a = 1.65$, $b = -0.68$ & $c = 0.13$. In all cases, parameters were chosen such that the stimulus had unit variance (and zero mean). The autocovariance of each stimulus, used to optimise neural responses, were computed analytically for each set of stimulus statistics. We added zero noise to the inputs (i.e. $X = Y$).

Neural responses were obtained by linearly filtering the stimulus, as described in the main text (with encoding filter of length 60). We optimised the encoding filters by maximising the IB objective function, separately for each stimulus. For panels 2c-d we used decoding filters of length $\tau = 1$; for panels 2e-f we used decoding filters of length $\tau = 60$. In each case we performed the optimisation with decoding lags ranging from $\Delta = 1$ to $\Delta = 10$, and a range of different bottleneck parameters, γ (which determined the channel capacity, C). SI fig. 2 plots the optimal encoding filters for each stimulus, with varying τ, C and Δ.

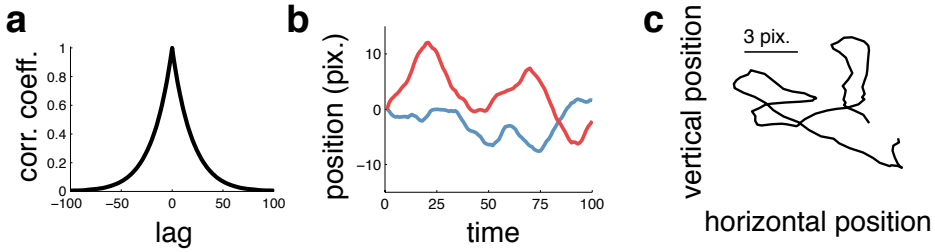

Figure 3: Motion trajectory of drifting natural image patches. (a) Autocorrelation of motion speeds along the x-axis. (b) Example trajectory in the x and y direction. (c) Full motion trajectory, in 2d.

3.2 Neural coding of naturalistic movie stimuli

For figure 3, we considered neural coding of naturalistic movie stimuli, consisting of stochastically drifting static images. Images were taken from the van Hateren natural image database (www.kyb.tuebingen.mpg.de/?id=227). Each image was normalised so that the pixels had zero mean and unit variance.

Each trial began with a 10×10 patch at a random position, $\{x_{\text{cord}}(0), y_{\text{cord}}(0)\}$, of the image. Movies were constructed by sliding the patch across the image. The position along the x-axis varied according to an AR2 process, described by: $x_{\text{cord}}(t) = x_{\text{cord}}(t-1) + v_x(t-1)$, where $v_x(t) = av_x(t-1) - b\eta(t)$, and $a = 0.95$ and $b = 0.16$ (see SI fig. 3), and η is a zero mean gaussian process. The position of the patch along the y-axis evolved according to the same dynamics. Trials where the patch reached the border of the image were excluded from the training data. The input to each neuron was created by adding gaussian white noise (with standard deviation of 0.1) to the stimulus.

We trained spatio-temporal encoding filters, W, with temporal length 3. We used a student-t approximation for the response distribution (see SI fig. 5 for comparison with results obtained using a gaussian approximation of the response distribution). Filters were initialised with uncorrelated white noise, of magnitude 10^{-2}. We first learned filters using with $\Delta = -6$. We then learned filters with increasing values of Δ. The encoding weights obtained for each Δ was used as the initial conditions for $\Delta + 1$. We also adjusted the bottleneck parameter, γ, so that the channel capacity remained constant across all Δ ($C \approx 32.5$bits).

To compute the 'directionality index' for each neuron, shown in fig. 3e, we first presented model neurons with drifiting sinusoidal grating stimuli, of varying phase, direction, and speed. We thus obtained the preferred phase/direction/speed for each model neuron that elicited strongest maximal response. The 'directionality index' was then computed for each neuron by comparing the neuron's maximum response to its preferred stimulus, and its response to a similar stimulus moving in the opposite direction.

3.3 Drifiting blob stimulli

For figure 4, we considered neural responses to 'drifting blob' stimuli, as shown in figure 4a. For this stimulus, there were 20 stimulus dimensions (i.e. 'pixels'), arranged along a single spatial axis. Note that to simplify our analysis we considered circular boundary conditions, so that each stimulus dimension corresponded to an angular coordinate, θ, arranged in equally spaced intervals between $-\pi$, and $\pi - 2\pi/20$.

Blob-like stimulus features were described by a (wrapped) gaussian, with standard deviation $\sigma_{blob} = 0.45$, time-varying position, $\theta_{\text{blob}}(t)$, and amplitude, $A(t)$. On each trial, the stimulus was constructed by adding two blob-like features, with varying amplitude and position. The position of each blob varied according to an AR2 process, according to: $\theta_{\text{blob}}(t) = \theta_{\text{blob}}(t-1) + v(t-1)$, where $v(t) = av(t-1) - b\eta(t)$, and $a = 0.90$ and $b = 0.14$. The ampltidue of each blob varied according to an AR1

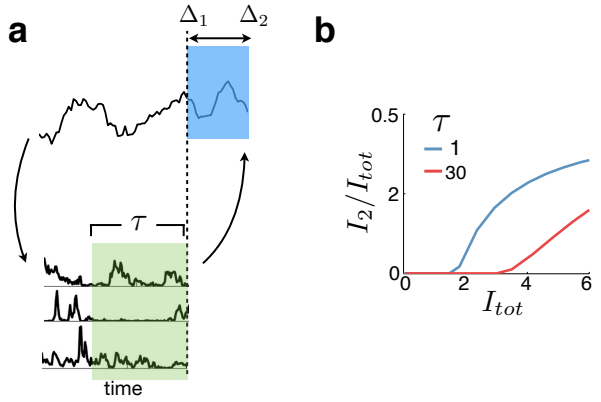

Figure 4: Schematic, where the goal is to reconstruct a stimulus in an extended temporal window (shaded blue). (b) We plotted the relative information encoded by two neurons about a 'two-timescale stimulus (fig. 2b) at varying channel capacity, C. The x-axis is the total information (or channel capacity) encoded by both neurons; the y–axis is the fraction of information encoded by the less active neuron. In blue we plot the results for an instantaneous code ($\tau = 0$); the red plot corresponds to a temporally extended code ($\tau = 30$). In both cases, at low channel capacity only one neuron is active; increasing the channel capacity above a certain threshold leads to the second neuron being active. The required threshold increases with the coding length, τ.

process: $A(t) = aA(t-1) + b\eta(t)$, where $a = 0.99$ and $b = 0.01$. The input to each neuron was created by adding gaussian white noise (with standard deviation of 0.2) to the stimulus.

Spatio-temporal encoding filters, W, were learned as for the naturalistic stimuli, with varying Δ, and γ chosen so that that the channel capacity remained constant across all Δ ($C = 8$ bits).

We sought to evaluate how well the neural responses encoded both the speed versus position the position of the stimulus after being optimised for efficient/predictve coding. To do this, we optimised responses with varying γ, and $\Delta = \pm 2$. We then computed neural responses to a stimulus, consisting of a single drifting blob feature, of constant amplitude. In each condition we computed the position/speed of the stimulus at time t from a linear readout of neural responses up until time t. We then computed the (circular) correlation between the reconstructed and true position/speed of the stimulus.

For fig. 4g we computed how different the response distribitution for each neuron was from a gaussian distribution (which we termed 'sparsity'). Specifically, we computed the 'negentropy' of each neuron's responses, defined as the difference between the entropy of a gaussian distribution (with variance equal to the neuron's response variance), and the entropy of each neuron's response (estimated empirically from the observed responses). This value is equal to zero if responses are gaussian distributed, and greater than zero otherwise.

For fig. 4h we computed the delay in responses to their preferred stimulus. For each neuron, we computed the preferred stimulus position, that generated maximal response (in the steady state). We then presented the moving bar stimulus and computed the average response of each neuron, before and after the stimulus is at its preferred location, given by: $u(\tau) = \sum_i r(t_i + \tau)$, where t_i denotes the i^{th} occurence of the stimulus at the neuron's preferred location. Finally, the 'delay' was defined as the weighted sum: $\sum_\tau |u(\tau)|\tau / \sum_\tau |u(\tau)|$.

4 Supplementary simulations

4.1 Number of neurons

For a given bottleneck parameter, γ, the IB algorithm automatically learns the appropriate number of neurons [40]. With a small channel capacity, C (i.e. large γ), all encoding weights, W, go to zero, so no information is encoded. As C is increased, neurons enter into the solution one-by-one.

The maximum number of neurons is determined by the dimensionality (and statistics) of the stimulus and target variable. Specifically, for the gaussian information bottleneck (i.e. $q(r)$, and $q(y|r)$ both gaussian), the number of neurons never exceeds the dimensionality of the decoded variable. In the main text the decoding window was always set to be one time-step long (see Fig. 1a). Therefore, the maximum number of neurons is equal to the stimulus dimensionality. For the simulations shown in figure 2, this means there is never more than a single neuron, irrespective of the channel capacity. This picture changes, however, when the decoding window is increased, so that the objective is to read-out the stimulus within a window, $[t + \Delta_1, t + \Delta_2]$ (SI equation 2). In this case, the maximum number of neurons is equal to the stimulus dimensionality multiplied by the decoding window length (SI fig. 2a).

To investigate what happens with a temporally extended decoding window, we performed the simulations shown in Fig. 2, with a decoding window extending between $t + 1$ and $t + 10$ time-steps. As in the main text, we began by considering an instantaneous code, with $\tau = 1$. In this case, the optimal coding weights can be expressed analytically [40]. With the Markov stimulus (Fig. 4b, upper panel), there is never more than one neuron, regardless of the channel capacity (because the future stimulus trajectory is fully determined by its current state). In contrast, with both the two-timescale and inertial stimulus (Fig. 2b, middle and lower panels), there is a maximum of two neurons (SI fig. 4b, blue). Interestingly, increasing the code length τ, reduces the number of neurons learned by the algorithm (SI fig. 4b, red). This is because increasing the code length, τ increases the dimension of the neural code, so that less neurons are required.

4.2 Comparison of gaussian and sparse IB results

In the main text we describe how we learned spatio-temporal encoding filters, to optimally encode naturalistic movies at different decoding lags, Δ (fig. 3). To do this, we approximated the response distribution, $q(r_t)$ using a student-t distribution (with shape parameter, ν learned from the data). When trained on data with a sparse latent structure, the IB algorithm learns a small shape parameter, ν, resulting in a heavy tailed, or 'sparse' response distribution, $q(r_t)$. Because of the similarity with traditional sparse coding models, we call this model 'sparse IB'.

SI fig. 5a plots the first 60 spatial filters (at 0-lag) learned with the sparse IB algorithm, with $\Delta = -6$, arranged in descending order of response magnitude. SI fig. 5b plots the mean squared filter response for each neuron ($\sigma_s^2 = \left\langle \left(\sum_k W_k x_{t-k}\right)^2 \right\rangle_t$), divided by the total response variance ($\sigma_s^2 + \sigma_n^2 = \left\langle \left(\sum_k W_k x_{t-k}\right)^2 \right\rangle_t + \Sigma_{ii}$).

We compared these results to the filters obtained with a 'gaussian IB' algorithm, where we approximated the response distribution, $q(r_t)$ using a gaussian distribution. SI fig. 5c plots the first 60 spatial filters (at 0-lag) obtained using this agorithm, with $\Delta = -6$. In contrast to the filters obtained with the sparse IB algorithm, the gaussian IB algorithm learns spatially non-local RFs (similar to the fourier decomposition of the image). SI fig. 5d plots the mean squared filter response for each neuron divided by its total response variance.

4.3 Parametric fit of spatiotemporal encoding filters

The spatial filters for each lag learned by the sparse IB algorithm were fitted using a 2d circular gabor function:

$$g(x, y) = \exp\left(\frac{1}{2\sigma^2}\left(x'^2 + y'^2\right)\right)\cos\left(2\pi\frac{x'}{\lambda} + \psi\right) \tag{10}$$

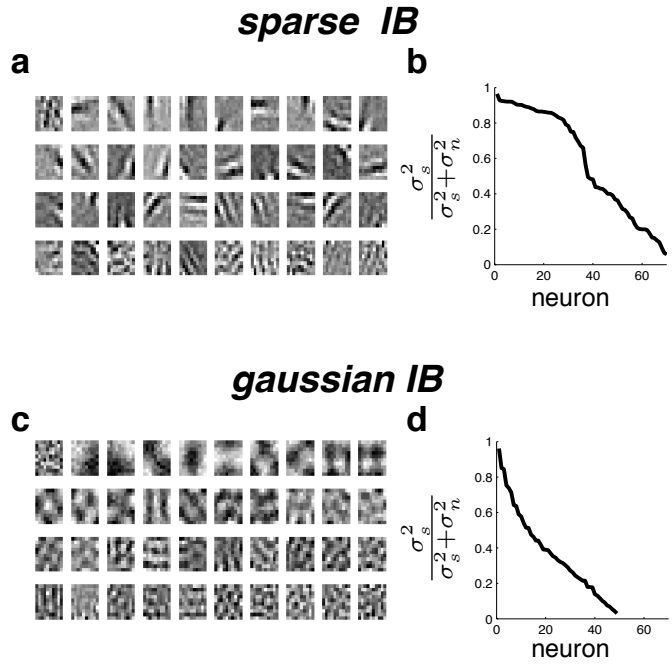

Figure 5: Comparison between sparse and gaussian IB. (a) First 60 spatial filters (at 0-lag) learned with the sparse IB algorithm, with $\Delta = -6$, arranged in descending order of response magnitude. (b) Mean squared filter response for each neuron (σ_s), divided by the total response variance ($\sigma_s^2 + \sigma_n^2$). (c-d) As for panels a-b, but with gaussian IB algorithm.

where

$$x' = (x - x0)\cos(\theta) + (y - y0)\sin(\theta) \quad (11)$$
$$y' = -(x - x0)\sin(\theta) + (y - x0)\cos(\theta). \quad (12)$$

Parameters were fitted using the 'autogaborsurf' function, written by Patrick Mineault, which evaluates the quality of fit for many different choices of parameters then refines the most promising set of parameters through least-squares (exhaustive search followed by refinement). We learned a different set of filters for the spatial filter at each lag. The average R2 goodness of fit was $R2 = 0.73$ for the model trained with $\Delta = -6$ and $R2 = 0.67$ for the model trained at $\Delta = 2$.

As described in the main text, when we trained the model with $\Delta = -6$, spatial filters at different lags resembled each other, though shifted in phase. To test this, we fitted gabor functions to the learned spatial filters at 0-lag, and then learned new values of ψ (determining the phase of the filter) for each lag (with all other parameters kept constant). We called this the 'phase shift' model. SI fig. 6a-b compares the spatio-temporal filters learned with the 'efficient coding' model (with $\Delta = -6$) for three example neurons with model fits obtained with the phase shift model. Supplementary figure 3c plots the mean squared error for each neuron obtained with the phase shift model, normalised by the mean squared error obtained when we fitted all the parameters of the gabor function at each lag. (Note that we discarded neurons from this analysis if the r2 value obtained with the full gabor model was less than 0.5). As can be seen, the phase shift model performed well at fitting the 'efficient coding' filters, with $\Delta = -6$, but poorly at fitting the 'predictive coding' filters, with $\Delta = 1$.

In contrast, 'predictive coding' filters, learned with $\Delta = 1$, were well fit by a spatio-temporally separable model, described by multiplying a spatial filter with a temporal filter. SI fig. 6d-e compares the spatio-temporal filters learned with the 'predictive coding' model (with $\Delta = 1$) for three example neurons with model fits obtained with the phase shift model. SI fig. 6f evaluates the mean squared

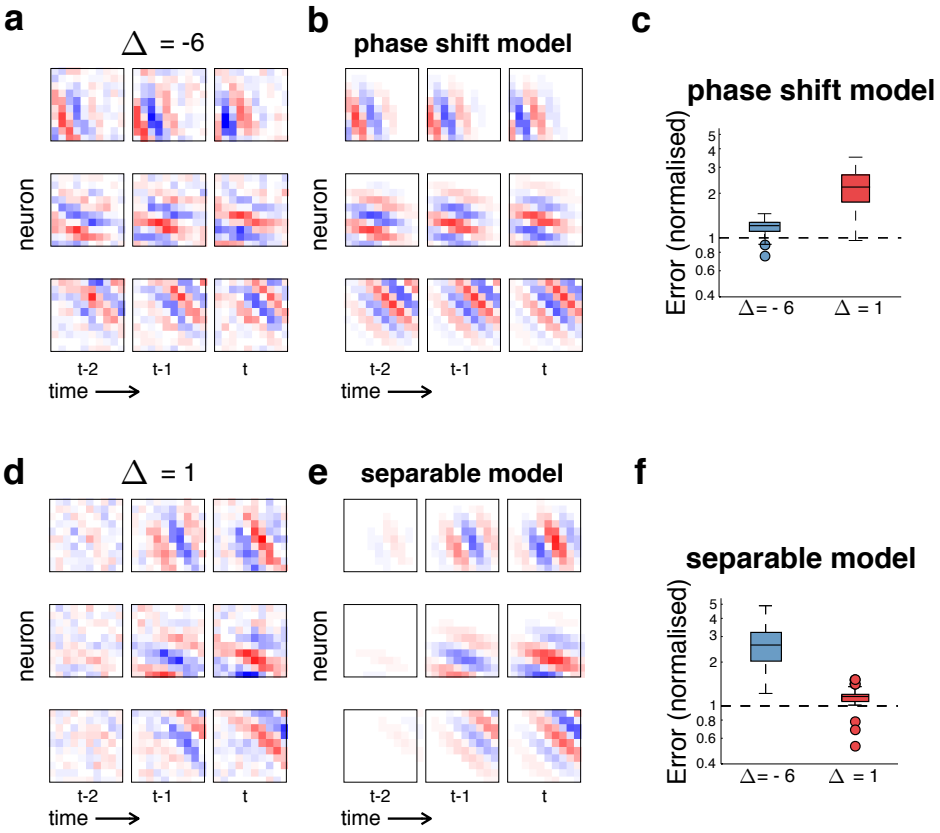

Figure 6: Parametric fits of encoding filters. (a) Spatio-temporal encoding filters, for 3 example neurons, after optimisation with $\Delta = -6$. Each row corresponds to a different neuron. Each column corresponds to the spatial filter at a given lag. (b) Parametric fits of encoding filters, using a circular gabor function. For the 'phase shift' model we fitted the parameters of the gabor filters to the spatial filter at 0-lag (i.e. the left column of panel a), and then varied the phase of the gabor filter to fit the spatial filter at each lag. (c) Mean squared error obtained with the phase-shift model for each neuron, divided by error obtained with the full model, where a seperate gabor filter was fitted to the encoding filters at each lag. Results are shown seperately for the filters trained at $\Delta = -6$ and $\Delta = 2$. (d) As for panel a, but with $\Delta = 1$. (e) Parametric fits of encoding filters, using a circular gabor function. For the 'seperable model', shown here, we fitted the parameters of the gabor filters to the spatial filter at 0-lag (i.e. the left column of panel a), and then varied the amplitude at each lag. (f) As for panel c, but for the separable model fits.

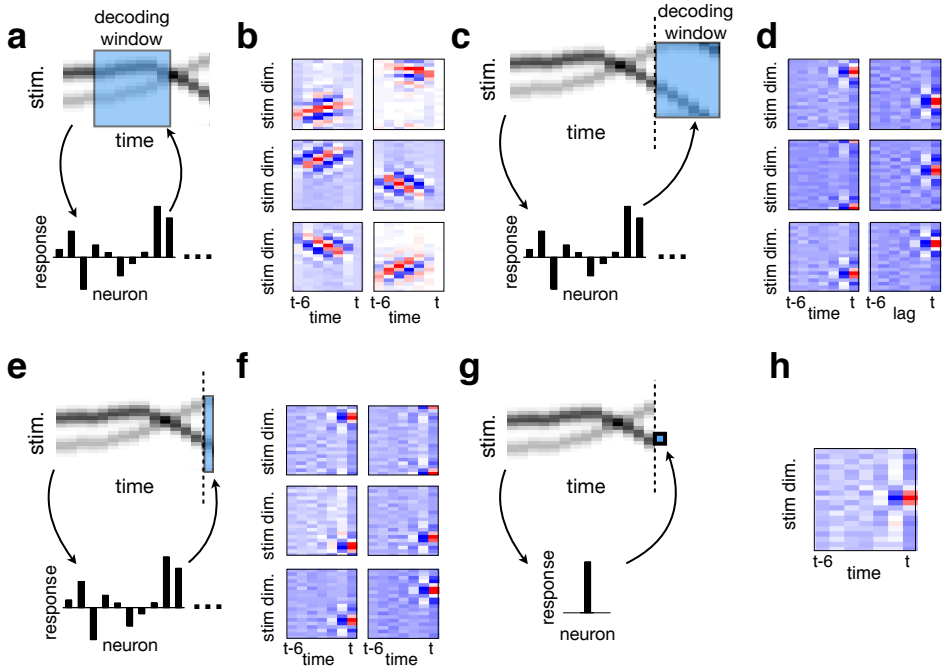

Figure 7: Motion tuning for efficient versus predictive coding. (a) 'Efficient coding' setup, where the objective is to reconstruct the stimulus in an extended temporal window (shown in blue) from the instantaneous neural responses. (b) Example spatiotemporal filters, obtained after training the network to reconstruct past stimuli, as in panel a. Each square corresponds to the filter for one neuron. (c) 'Predictive coding' setup, where the objective is to reconstruct the stimulus in an extended temporal window (shown in blue) in the *future*. (d) As for panel b, but for the predictive coding setup, shown in panel c. (e) Alternative predictive coding set-up, where the decoding window consists of only one time-step in the future. (e) Encoding filters obtained after training the model with the setup shown in panel e. (g) Reduced set-up, where the decoding window consists of a single pixel and single time-step in the future. (h) Encoding filter obtained after training single neuron on the setup shown in panel g.

error for each neuron obtained with a separable model, normalised by the mean squared error obtained when we fitted all the parameters of the gabor function at each lag. (Again we discarded neurons from this analysis if their r2 value obtained with the full gabor model was less than 0.5). In this case, the separable model performed poorly at fitting the 'efficient coding' spatial filters, with $\Delta = -6$, but could well fit the 'predictive coding' filters, with $\Delta = 1$.

4.4 Why does predictive coding not result in motion filters?

In the main text, we showed, using drifting blob stimuli, that efficient coding (i.e. with $\Delta \ll 0$) results in motion filters, while predictive coding (i.e. with $\Delta > 0$) does not. To understand why this is the case, we considered the set-up shown in Supplementary figure 4a & c where neural responses in a single time window (i.e. $\tau = 0$) are used to reconstruct the stimulus in an extended time window (represented in the figure by a blue square), which can be either in the past (as in SI fig. 7a) or future (as in SI fig. 7c).

When we optimised neural encoding filters to perform 'efficient coding' (i.e. with the decoding window in the past, as in SI fig. 7a), neurons learned to respond to motion in a particular direction (SI fig. 7b). This was expected, as such a code allows neurons to respond as sparsely as possible, and thus achieve an efficient representation of presented stimuli.

Table 1: Factors determining the optimal neural code. First three explored in main text.

Factor	Control parameter	Consequence
Coding capacity	C	Fig 2
Decoding window	τ	Fig 2
Decoding lag	Δ	Fig 2-4
Input noise magnitude	noise n, added to stim. y	SI Appendix 4.5.1
Temp. corr. in spiking	2^{nd} term in Eq (1)	SI Appendix 4.5.2
Stim. prediction window	$Y_{(t+\Delta_1:t+\Delta_2)}$ in Eq (1)	SI Appendix 4.5.3
Encoding model	parametric form of $p(r\|x)$	SI Appendix 4.5.4

In contrast, when we optimised neural encoding filters for 'predictive coding' (i.e. with the decoding window in the future, as in SI fig. 4c), neurons were no longer selective for a particular motion direction (SI fig. 7f). We hypothesized that this occurs because, when the decoding window is in the future, the algorithm prioritises reconstructing stimuli at the beginning of the decoding window, as these are the most predictable. To test this, we reran the optimisation algorithm with a short decoding window, consisting of a single time-step (SI fig. 7e). In support of our hypothesis, the resulting encoding filters were qualitatively unaltered (SI fig. 7f).

Assuming that the algorithm allocates all coding resources to reconstructing the stimulus at a single time-step in the future, one can show that the resulting encoding filters will be unselective to the direction of the stimulus. To see this, first consider the case where each neuron encodes a single spatial location (SI fig. 7g). Because, in our simulations, stimuli move in both directions equally often, each neuron will not be selective to a particular direction; the stimulus is equally likely to approach the encoded location from either direction. Thus, the resulting spatiotemporal RF for one such neuron is shown in SI fig. 7h. Moreover, this argument holds even if neurons are selective for stimuli at multiple locations; linear combinations of filters of the form shown in SI fig. 7h, centred on different locations, cannot result in motion filters.

Note that, the set-up described in the main text, where a temporally distributed neural code (i.e. with $\tau > 0$) is used to reconstruct the stimulus at single time-point, $t+\Delta$ (see Fig. 1a), differs from the set-up considered here (illustrated in SI fig. 7c). However, the arguments underlying why we observe motion filters are similar in both cases.

4.5 Dependence of neural code on constraints and functional goals

In the main text we investigated how the optimal neural code depends on the functional goals/constraints of the system. Specifically, we looked at how the code varies depending on: the decoding lag, Δ, code length τ, and channel capacity C (fig. 1b). In addition, our framework allows us to vary several other factors, which each alter the optimal code in different ways (SI Table 1).

4.5.1 Input noise

In our simulations we added gaussian white noise to the stimulus, Y, to generate the noisy sensory input, X. Varying the magnitude of this input noise alters the resulting code, as neurons have to smooth over their sensory input to encode information about the stimulus. Increasing the input noise has a similar effect to decreasing the coding capacity, resulting in larger neural RFs, and reducing the degree to which neurons decorrelate the input signal. With the 'moving gaussian blob' stimulus, shown in figure 4 of the main text, increasing the input noise results in neurons that are not direction selective (SI fig. 8b).

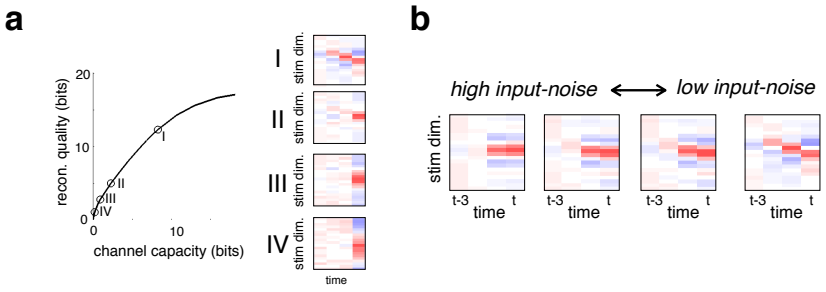

Figure 8: Effect of varying the internal and external noise. (a) Information encoded about the 'gaussian bump' stimulus (at lag $\Delta = -2$), shown in fig. 4 of the main text, versus total channel capacity, C (i.e. variable internal noise). Spatiotemporal encoding filters for an example neuron at various different channel capacities (denoted by open circles information curve) are shown to the right. (b) Spatiotemporal filters obtained with varying amplitude of the external noise, added to the input.

4.5.2 Constraint on neural correlations

As shown in SI equation 1, in a straightforward application of information bottleneck framework we would constrain the information between neural responses in a time window $t - \tau$ to t, and stimulus up until time t: $I(R_{t-\tau:t}; X_{-\infty:t})$. However, in order to compare our framework directly with previous work on efficient coding, we instead constrained the information between the instantaneous neural responses at time t and the stimulus up until time t: $I(R_t; X_{-\infty:t})$ (see SI equation 2). In this case, in the limit where $X \to Y$ and $\gamma \to 1$, maximising the IB objective function is equivalent to minimising the redundancy between neural responses at different time-steps, as in standard efficient coding.

In contrast, if we constrain the full mutual information $I(R_{t-\tau:t}; X_{-\infty:t})$, rather than just depending on the marginal response statistics of each neuron, the constraint term depends on the spatiotemporal dependencies between responses of different neurons & at different times. With a gaussian stimulus, as in fig. 2, this corresponds to constraining the full spatiotemporal response covariance matrix, rather than just the response variance of each neuron. We found that this generally results in responses that are more spatiotemporally correlated (i.e. more 'smooth'), compared to the results shown in the main text (SI fig. 9).

4.5.3 Length of decoding window

In the main text, we considered the case where the objective is to reconstruct the stimulus at a single time-point, $t+\Delta$. More generally, we can consider the case where the goal is to reconstruct the stimulus in an extended window of variable length (as in SI fig. 4 & 7). However, for the simulations shown in figure 2 of the main text increasing the length of the decoding window had little qualitative effect on neural responses, (except at the limit of very high channel capacity where it resulted in an additional neuron being recruited; SI fig. 4). Likewise, for the sparse stimuli shown in Fig. 3-4, increasing the length of the decoding window had little qualitative effect, as the IB algorithm dedicated the majority of its resources to encoding the stimulus at the beginning of the decoding window, where the stimulus is the most predictable (e.g. compare SI fig. 7c and f).

4.5.4 Constraint on encoding model

In the main text we considered a linear encoding model. As described in the discussion, further extensions to our work could treat non-linear encoding models. The qualitative effect of using a non-linear encoding model will vary greatly depending on the stimulus statistics. For example, we previously showed how to combine our variational IB framework with a non-linear kernel encoding model (Chalk et al. NIPS 2016). Here we found that using such a non-linear model only had a

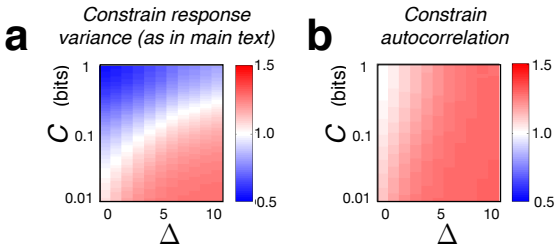

Figure 9: Correlation index optimisation with varying Δ & C. (a) Correlation index obtained with varying C and Δ after optimising the neural code towards the 2-timescale stimulus, shown in figure 2b of the main text. (b) As in panel a, but where we constrain the correlations in the neural response, as described in SI section 3.5.2.

strong qualitative effect on neural responses when trained on a complex non-linear task (such as image completion), but not when trained on a simple task (such as image denoising). Future work will be required to investigate what are the consequnces of such a non-linear model on predictive coding.

References

[1] F. Attneave (1954), Some informational aspects of visual perception. Psychol. Rev. 61(3):183–93.

[2] R. Linsker (1988), Deriving receptive fields using an optimal encoding criterion. IEEE Computer, 21, 105–17.

[3] R. Linsker, (1989). An application of the principle of maximum information preservation to linear systems. In "Advances in neural information processing systems" 186-194.

[4] J. H. van Hateren, (1992). A theory of maximizing sensory information. Biological cybernetics, 68(1):23–29.

[5] H. B. Barlow (1961), Possible principles underlying the transformation of sensory messages. In Sensory Communication, ed. WA Rosenblith, pp. 217–34, Cambridge, MA: MIT Press.

[6] E. P. Simoncelli, B. A. Olshausen (2001), Natural image statistics and neural representation. Annu. Rev. Neurosci. 24:1193–216

[7] H. Barlow (2001) Redundancy reduction revisited. Network: computation in neural systems 12(3): 241–253.

[8] D. J. Field (1987), Relations between the statistics of natural images and the response properties of cortical cells. J Opt Soc Am A 4: 2379–2394.

[9] D. Kersten (1987), Predictability and redundancy of natural images. J Opt Soc Am A 4: 2395–2400.

[10] M. V. Srinivasan, S. B. Laughlin, & A. Dubs (1982). Predictive coding: a fresh view of inhibition in the retina. Proc of the Royal Society of London B: Biological Sciences, 216(1205):427–459.

[11] Y. Dan, J. J. Atick, & R. C. Reid, R. C. (1996). Efficient coding of natural scenes in the lateral geniculate nucleus: experimental test of a computational theory. J Neurosci, 16(10):3351-3362.

[12] B. A. Olshausen & D. J. Field (2004), Sparse coding of sensory inputs. Curr. Op. in Neurobio. 14(4):481–487.

[13] A. Hyvärinen, J. Hurri & P. O. Hoyer (2009) Natural Image Statistics; a probabilistic approach to early computational vision. Springer-Verlag.

[14] A. J. Bell, & T. J. Sejnowski, (1995). An information-maximization approach to blind separation and blind deconvolution. Neural computation, 7(6):1129–1159.

[15] A. J. Bell, & T. J. Sejnowski, (1997). The independent components of natural scenes are edge filters. Vision research, 37(23):3327–3338.

[16] B. A. Olshausen & D. J. Field (1996), Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature, 381(6583):607–609.

[17] J. H. van Hateren, A. van der Schaaf (1998), Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex. Proc. R. Soc. London Ser. B, 265:359–66.

[18] E. C. Smith & M. S. Lewicki (2006), Efficient auditory coding. Nature, 439(7079):978–982.

[19] F. E. Theunissen (2003), From synchrony to sparseness. Trends Neurosi, 26:61–64

[20] R. P. Rao, & D. H. Ballard (1999), Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. Nature neuroscience, 2(1):79–87.

[21] M. Boerlin, & S. Deneve (2011), Predictive coding of dynamical variables in balanced spiking networks. PLoS Comput Biol, 7(2), e1001080.

[22] S. Druckmann, T. Hu & D. B. Chklovskii (2017), A mechanistic model of early sensory processing based on subtracting sparse representations. in Advances in Neural Information Processing Systems 25:1979–1987.

[23] J. J. Atick & A. N. Redlich (1992), What does the retina know about natural scenes? Neural Computation, 4(2):196–210.

[24] E. Doi, & M. S. Lewicki (2005), Sparse Coding of Natural Images Using an Overcomplete Set of Limited Capacity Units. Advances in Neural Information Processing Systems 17.

[25] E. Doi, M. S. Lewicki (2014), A simple model of optimal population coding for sensory systems. PLOS Comp. Bio., 10(8):e1003761

[26] Y. Karklin & E. P. Simoncelli (2011), Efficient coding of natural images with a population of noisy Linear-Nonlinear neurons. Advances in Neural Information Processing Systems 24, pp. 999–1007.

[27] B. A. Brinkman., A. I. Weber, F. Rieke, & E. Shea-Brown (2016), How Do Efficient Coding Strategies Depend on Origins of Noise in Neural Circuits? PLoS Comp Bio, 12(10):e1005150.

[28] G. Tkačik, J. S. Prentice, V. Balasubramanian,& E. Schneidman (2010), Optimal population coding by noisy spiking neurons. Proc. of the National Academy of Sciences, 107(32):14419-24.

[29] N. Brenner, W. Bialek, & R. de R. van Steveninck (2000) Adaptive rescaling maximizes information transmission. Neuron, 26(3):695–702

[30] A. L. Fairhall, G. D. Lewen, W. Bialek, W., & R. de R. van Steveninck (2001). Efficiency and ambiguity in an adaptive neural code. Nature, 412(6849):787–792

[31] C.. K.. Machens, T. Gollisch, O. Kolesnikova, & A. V. M. Herz (2005), Testing the efficiency of sensory coding with optimal stimulus ensembles. Neuron, 47(3):447–456.

[32] R. Desimone. (1991) Face-selective cells in the temporal cortex of monkeys. J Cog Neurosci 3(1):1-8.

[33] W. Bialek, R. De Ruyter Van Steveninck, & N. Tishby (2006), Efficient representation as a design principle for neural coding and computation. IEEE International Symposium on Information Theory, pp. 659–663.

[34] J. Salisbury & S. Palmer (2016), Optimal prediction in the retina and natural motion statistics. Journal of Statistical Physics, 162(5):1309–1323.

[35] S. E. Palmer, O. Marre, M. J. Berry II, & W. Bialek (2015), Predictive information in a sensory population. Proc Natl Acad Sci USA, 112(22):6908–6913.

[36] L. Buesing & W. Maass (2010), A spiking neuron as information bottleneck. Neural Comp., 22(8):1961–1992..

[37] F. Creutzig & H. Sprekeler (2008), Predictive coding and the slowness principle: An information-theoretic approach. Neural Comp, 20:1026–1041.

[38] P. Berkes & L. Wiskott (2005), Slow feature analysis yields a rich repertoire of complex cell properties. J. Vis. 5(6).

[39] J. P. Lies, R. M. Hafner, & M. Bethge, (2014), Slowness and sparseness have diverging effects on complex cell learning. PLoS Comp Bio, 10(3):e1003468.

[40] N. Tishby, F. C. Pereira, & W. Bialek (1999), The information bottleneck method. Proc. of the 37th Annual Allerton Conference on Communication, Control and Computing, pp. 368–377.

[41] M. Chalk, O. Marre, & G. Tkačik (2016), Relevant sparse codes with variational information bottleneck. Advances in Neural Information Processing Systems, 1957–1965